Kathleen Sullivan (SBN 242261)
kathleensullivan@quinnemanuel.com
QUINN EMANUEL URQUHART & SULLIVAN LLP
51 Madison Avenue, 22nd Floor
New York, NY 10010
Telephone: (212) 849-7000
Facsimile: (212) 849-7100

Sean S. Pak (SBN 219032)
seanpak@quinnemanuel.com
John M. Neukom (SBN 275887)
johnneukom@quinnemanuel.com.
QUINN EMANUEL URQUHART & SULLIVAN LLP
50 California Street, 22nd Floor
San Francisco, CA 94111
Telephone: (415) 875-6600
Facsimile: (415) 875-6700

Mark Tung (SBN 245782)
marktung@quinnemanuel.com
QUINN EMANUEL URQUHART & SULLIVAN LLP
555 Twin Dolphin Drive, 5th Floor
Redwood Shores, CA 94065
Telephone: (650) 801-5000
Facsimile: (650) 801-5100

Steven Cherny *(admission pro hac vice pending)*
steven.cherny@kirkland.com
KIRKLAND & ELLIS LLP
601 Lexington Avenue
New York, New York 10022
Telephone: (212) 446-4800
Facsimile: (212) 446-4900

Adam R. Alper (SBN 196834)
adam.alper@kirkland.com
KIRKLAND & ELLIS LLP
555 California Street
San Francisco, California 94104
Telephone: (415) 439-1400
Facsimile: (415) 439-1500

Michael W. De Vries (SBN 211001)
michael.devries@kirkland.com
KIRKLAND & ELLIS LLP
333 South Hope Street
Los Angeles, California 90071
Telephone: (213) 680-8400
Facsimile: (213) 680-8500

Attorneys for Plaintiff Cisco Systems, Inc.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA, SAN JOSE DIVISION

CISCO SYSTEMS, INC.,

Plaintiff,

vs.

ARISTA NETWORKS, INC.,

Defendant.

CASE NO. 5:14-cv-5344-BLF

DECLARATION OF KEVIN C. ALMEROTH SUBMITTED IN SUPPORT OF PLAINTIFF CISCO SYSTEMS, INC.'S OPENING CLAIM CONSTRUCTION BRIEF

I. INTRODUCTION

1. I have been retained by counsel for Plaintiff Cisco Systems, Inc. ("Plaintiff" or "Cisco") as an expert in this litigation to provide opinions regarding certain claim terms in U.S. Patent Nos. 7,047,526 ("the '526 patent"), and 7,953,886 ("the '886 patent") (collectively "the patents"). In particular, I have been asked to analyze what a person of ordinary skill in the art would understand certain claims terms to mean at the time of invention for each of the above-mentioned patents.

2. I have been engaged by counsel for Cisco. I am being compensated by counsel for Cisco at an hourly rate of $600. My compensation does not depend in any way on the outcome of this investigation or the particular testimony or opinions I provide.

3. In rendering my opinions, I considered the items listed in Appendix A attached to this Declaration, the items discussed or listed herein, as well as my own experiences in the field. I understand that discovery in this litigation is still ongoing, and I reserve the right to amend or supplement my opinions in light of further documents, depositions, or discovery disclosures. I further reserve the right to rely upon any additional information or materials that may be provided to me or that are relied upon by any of Arista's experts or witnesses, if called to testify or to give additional opinions regarding this matter.

II. QUALIFICATIONS AND EXPERIENCE

4. I summarize in this section my educational background, career history, publications, and other relevant qualifications.

5. I hold three degrees from the Georgia Institute of Technology: (1) a Bachelor of Science degree in Information and Computer Science (with minors in Economics, Technical Communication, American Literature) earned in June, 1992; (2) a Master of Science degree in Computer Science (with specialization in Networking and Systems) earned in June, 1994; and (3) a

Doctor of Philosophy (Ph.D.) degree in Computer Science (Dissertation Title: Networking and System Support for the Efficient, Scalable Delivery of Services in Interactive Multimedia System, minor in Telecommunications Public Policy) earned in June, 1997.

6. One of the major themes of my research has been the delivery of multimedia content and data between computing devices and users. In my research I have looked at large-scale content delivery systems and the use of servers located in a variety of geographic locations to provide scalable delivery to hundreds, even thousands, of users simultaneously. I have also looked at smaller-scale content delivery systems in which content, including interactive communication like voice and video data, is exchanged between computers and portable computing devices. As a broad theme, my work has examined how to exchange content more efficiently across computer networks, including the devices that switch and route data traffic. More specific topics include the scalable delivery of content to many users, mobile computing, satellite networking, delivering content to mobile devices, and network support for data delivery in wireless network.

7. Beginning in 1992, when I started graduate school, the first focus of my research was on the provision of interactive functions (VCR-style functions like pause, rewind, and fast-forward) for near video-on-demand systems in cable systems, in particular, how to aggregate requests for movies at a cable head-end and then how to satisfy a multitude of requests using one audio/video stream broadcast to multiple receivers simultaneously. Continued evolution of this research has resulted in the development of new techniques to scalably deliver on-demand content, including audio, video, web documents, and other types of data, through the Internet and over other types of networks, including over cable systems, broadband telephone lines, and satellite links.

8. An important component of my research from the very beginning has been investigating the challenges of communicating multimedia content between computers and across networks. Although the early Internet was designed mostly for text-based non-real time

applications, the interest in sharing multimedia content quickly developed. Multimedia-based applications ranged from downloading content to a device to streaming multimedia content to be instantly used. One of the challenges was that multimedia content is typically larger than text-only content but there are also opportunities to use different delivery techniques since multimedia content is more resilient to errors. I have worked on a variety of research problems and used a number of systems that were developed to deliver multimedia content to users.

9. In 1994, I began to research issues associated with the development and deployment of a one-to-many communication facility (called "multicast") in the Internet (first deployed as the Multicast Backbone, a virtual overlay network supporting one-to-many communication). Some of my more recent research endeavors have looked at how to use the scalability offered by multicast to provide streaming media support for complex applications like distance learning, distributed collaboration, distributed games, and large-scale wireless communication. Multicast has also been used as the delivery mechanism in systems that perform local filtering (i.e., sending the same content to a large number of users and allowing them to filter locally content in which they are not interested).

10. Starting in 1997, I worked on a project to integrate the streaming media capabilities of the Internet together with the interactivity of the web. I developed a project called the Interactive Multimedia Jukebox (IMJ). Users would visit a web page and select content to view. The content would then be scheduled on one of a number of channels, including delivery to students in Georgia Tech dorms delivered via the campus cable plant. The content of each channel was delivered using multicast communication.

11. In the IMJ, the number of channels varied depending on the capabilities of the server including the available bandwidth of its connection to the Internet. If one of the channels was idle, the requesting user would be able to watch their selection immediately. If all channels were

streaming previously selected content, the user's selection would be queued on the channel with the shortest wait time. In the meantime, the user would see what content was currently playing on other channels, and because of the use of multicast, would be able to join one of the existing channels and watch the content at the point it was currently being transmitted.

12. The IMJ service combined the interactivity of the web with the streaming capabilities of the Internet to create a jukebox-like service. It supported true Video-on-Demand when capacity allowed, but scaled to any number of users based on queuing requested programs. As part of the project, we obtained permission from Turner Broadcasting to transmit cartoons and other short-subject content. We also attempted to connect the IMJ into the Georgia Tech campus cable television network so that students in their dorms could use the web to request content and then view that content on one of the campus's public access channels.

13. More recently, I have also studied issues concerning how users choose content, especially when considering the price of that content. My research has examined how dynamic content pricing can be used to control system load. By raising prices when systems start to become overloaded (i.e., when all available resources are fully utilized) and reducing prices when system capacity is readily available, users' capacity to pay as well as their willingness can be used as factors in stabilizing the response time of a system. This capability is particularly useful in systems where content is downloaded or streamed to users on-demand.

14. As a parallel research theme, starting in 1997, I began researching issues related to wireless devices. In particular, I was interested in showing how to provide greater communication capability to "lightweight devices," i.e., small form-factor, resource-constrained (e.g., CPU, memory, networking, and power) devices.

15. Starting in 1998, I published several papers on my work to develop a flexible, lightweight, battery-aware network protocol stack. The lightweight protocols we envisioned were

similar in nature to protocols like Universal Plug and Play (UPnP) and Digital Living Network Alliance (DLNA).

16. From this initial work, I have made wireless networking-including ad hoc and mesh networks and wireless devices-one of the major themes of my research. One topic includes developing applications for mobile devices, for example, virally exchanging and tracking "coupons" through "opportunistic contact" (i.e., communication with other devices coming into communication range with a user). Other topics include building network communication among a set of mobile devices unaided by any other kind of network infrastructure. Yet another theme is monitoring wireless networks, in particular different variants of IEEE 802.11 compliant networks, to (1) understand the operation of the various protocols used in real-world deployments, (2) use these measurements to characterize use of the networks and identify protocol limitations and weaknesses, and (3) propose and evaluate solutions to these problems.

17. As an important component of my research program, I have been involved in the development of academic research into available technology in the market place. One aspect of this work is my involvement in the Internet Engineering Task Force (IETF) including many content delivery-related working groups like the Audio Video Transport (AVT) group, the MBone Deployment (MBONED) group, Source Specific Multicast (SSM) group, the Inter- Domain Multicast Routing (IDMR) group, the Reliable Multicast Transport (RMT) group, the Protocol Independent Multicast (PIM) group, etc. I have also served as a member of the Multicast Directorate (MADDOGS), which oversaw the standardization of all things related to multicast in the IETF. Finally, I was the Chair of the Internet2 Multicast Working Group for seven years.

18. I am an author or co-author of nearly 200 technical papers, published software systems, IETF Internet Drafts and IETF Request for Comments (RFCs).

19. My involvement in the research community extends to leadership positions for several journals and conferences. I am the co-chair of the Steering Committee for the ACM Network and System Support for Digital Audio and Video (NOSSDAV) workshop and on the Steering Committees for the International Conference on Network Protocols (ICNP), ACM Sigcomm Workshop on Challenged Networks (CHANTS), and IEEE Global Internet (GI) Symposium. I have served or am serving on the editorial boards of IEEE/ACM Transactions on Networking, IEEE Transactions on Mobile Computing, IEEE Transactions on Networks and System Management, IEEE Network, ACM Computers in Entertainment, AACE Journal of Interactive Learning Research (JILR), and ACM Computer Communications Review.

20. I have co-chaired a number of conferences and workshops including the IEEE International Conference on Network Protocols (ICNP), ACM International Conference on Next Generation Communication (CoNext), IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), International Conference on Communication Systems and Networks (COMSNETS), IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS), the International Workshop On Wireless Network Measurement (WiNMee), ACM Sigcomm Workshop on Challenged Networks (CHANTS), the Network Group Communication (NGC) workshop, and the Global Internet Symposium; and I have been on the program committee of numerous conferences.

21. Furthermore, in the courses I teach, the class spends significant time covering all aspects of the Internet including each of the layers of the Open System Interconnect (OSI) protocol stack commonly used in the Internet. These layers include the physical and data link layers and their handling of signal modulation, error control, and data transmission. I also teach DOCSIS, DSL, and other standardized protocols for communicating across a variety of physical media including cable systems, telephone lines, wireless, and high-speed Local Area Networks (LANs).

I teach the configuration and operation of switches, routers, and gateways including routing and forwarding and the numerous respective protocols as they are standardized and used throughout the Internet. Topics include a wide variety of standardized Internet protocols at the Network Layer (Layer 3), Transport Layer (Layer 4), and above.

22. In addition to having co-founded a technology company myself, I have worked for, consulted with, and collaborated with companies such as IBM, Hitachi Telecom, Digital Fountain, RealNetworks, Intel Research, Cisco Systems, and Lockheed Martin.

23. I am a Member of the Association of Computing Machinery (ACM) and a Fellow of the Institute of Electrical and Electronics Engineers (IEEE).

24. I attach as Appendix B my *curriculum vitae*¸ which includes a complete list of my qualifications.

III. LEGAL STANDARDS

25. I understand that claim construction is for the Court to decide. I further understand that claim terms should be given their ordinary and customary meaning within the context of the patent in which the terms are used, i.e., the meaning that the term would have to a person of ordinary skill in the art in question at the time of the invention in light of what the patent teaches.

26. I understand that to determine how a person of ordinary skill would understand a claim term, one should look to those sources available that show what a person of skill in the art would have understood disputed claim language to mean. Such sources include the words of the claims themselves, the remainder of the patent's specification, the prosecution history of the patent (all considered "intrinsic" evidence), and "extrinsic" evidence concerning relevant scientific principles, the meaning of technical terms, and the state of the art.

27. I understand that words or terms should be given their ordinary and accepted meaning unless it appears that the inventors were using them to mean something else. In making this

determination, however, of paramount importance are the claims, the patent specification, and the prosecution history. Additionally, the specification and prosecution history must be consulted to confirm whether the patentee has acted as its own lexicographer (i.e., provided its own special meaning to any disputed terms), or intentionally disclaimed, disavowed, or surrendered any claim scope.

28. I understand that the claims of a patent define the scope of the rights conferred by the patent. The claims particularly point out and distinctly claim the subject matter which the patentee regards as his invention. Because the patentee is required to define precisely what he claims his invention to be, it is improper to construe claims in a manner different from the plain import of the terms used consistent with the specification. Accordingly, a claim construction analysis must begin and remain centered on the claim language itself. Additionally, the context in which a term is used in the asserted claim can be highly instructive. Likewise, other claims of the patent in question, both asserted and unasserted, can inform the meaning of a claim term. For example, because claim terms are normally used consistently throughout the patent, the usage of a term in one claim can often illuminate the meaning of the same term in other claims. Differences among claims can also be a useful guide in understanding the meaning of particular claim terms.

29. I understand that a person of ordinary skill in the art is deemed to read a claim term not only in the context of the particular claim in which the disputed term appears, but in the context of the entire patent, including the specification. For this reason, the words of the claim must be interpreted in view of the entire specification. The specification is the primary basis for construing the claims and provides a safeguard such that correct constructions closely align with the specification. Ultimately, the interpretation to be given a term can only be determined and confirmed with a full understanding of what the inventors actually invented and intended to envelop with the claim as set forth in the patent itself.

30. I understand that claim terms must be construed in a manner consistent with the context of the intrinsic record. In addition to consulting the specification, one should also consider the patent's prosecution history. The prosecution file history provides evidence of how both the Patent Office and the inventors understood the terms of the patent, particularly in light of what was known in the prior art. Further, where the specification describes a claim term broadly, arguments and amendments made during prosecution may require a more narrow interpretation.

31. I understand that while intrinsic evidence is of primary importance, extrinsic evidence, e.g., all evidence external to the patent and prosecution history, including expert and inventor testimony, dictionaries, and learned treatises, can also be considered. For example, technical dictionaries may help one better understand the underlying technology and the way in which one of skill in the art might use the claim terms.

32. I understand a patent is invalid for indefiniteness if its claims, read in light of the specification delineating the patent, and the prosecution history, fail to inform, with reasonable certainty, those skilled in the art about the scope of the invention. A claim term is indefinite if a person of skill in the art could not discern the "metes and bounds" of the claim.

IV. THE '886 PATENT

33. The '886 patent, titled "Method and System for Receiving and Translating CLI Command Data Within a Routing System," was filed July 8, 2005. The '886 patent issued on May 31, 2011.

34. Generally speaking, the '886 patent relates to "routing systems for computer networks, and more particularly to the transmission of instructions to and receipt of data from such routing systems." '886 Patent at 1:8-10. According to the '886 patent, and as it was well understood in the art, access for routers was generally accomplished via human input at the "command line interface," also known "CLI." *Id.* at 1:14-16. The '886 patent identifies that a

problem with the CLI interface was that while it was comprehensive, it was difficult and cumbersome to automate. *Id.* at 1:32-33.

35. To remedy this problem, the ‘886 patent discloses a system and method that allows for a more structured approach for accessing routers. *Id.* at 1:34-35. In one embodiment of the disclosed invention described in the specification of the ‘886 patent, CLI command input statements and output statements are formatted in accordance with an XML schema of CLI rules and behaviors. *Id.* at 3:22-26. Specifically, the ‘886 patent discloses an embodiment where a CLI parser subsystem of the routing system translates the input commands from XML to CLI. The ‘886 patent further discloses that the input commands are configured in, for example, an XML format having a CLI syntax. *Id.* at 4:4-32. Similarly, the CLI parser subsystem can also translate CLI output statements from CLI into a different format such as XML. *Id.* at 4:33-42.

A. <u>LEVEL OF ORDINARY SKILL IN THE ART</u>

36. I am familiar with the knowledge of a person of ordinary skill in the field related to the subject matter of the ‘886 patent in the 2005 time period. In forming my opinion, I considered several things, including the various approaches to input/output interfaces of routing systems employed in the art, the type of problems encountered, and the rapidity with which innovations were made. I also considered the sophistication of the technology involved, and the educational background and experience of those actively working in the field. Finally, I placed myself back in the 2005 time frame, and considered the engineers that I had taught and worked with in the networking industry. For the purposes of this declaration, I am of the opinion that a person of ordinary skill in the art with respect to the patents would have a Bachelor’s of science degree in electrical engineering, computer science or engineering, or a related field, and two to four years of work or research experience in the field of networking or information networks, or a Master’s degree and one to two years of experience.

37. I meet these criteria and consider myself a person with at least ordinary skill in the art pertaining to the '886 patent. I was such a person by at least 2005.

38. I understand Cisco has asserted that the claims of the '886 patent is entitled to a priority date of December 3, 2003. My opinions with respect to the level of ordinary skill are also applicable to the 2003 time frame.

B. CLAIM TERMS OF THE '886 PATENT

1. "extensible markup language (XML)"

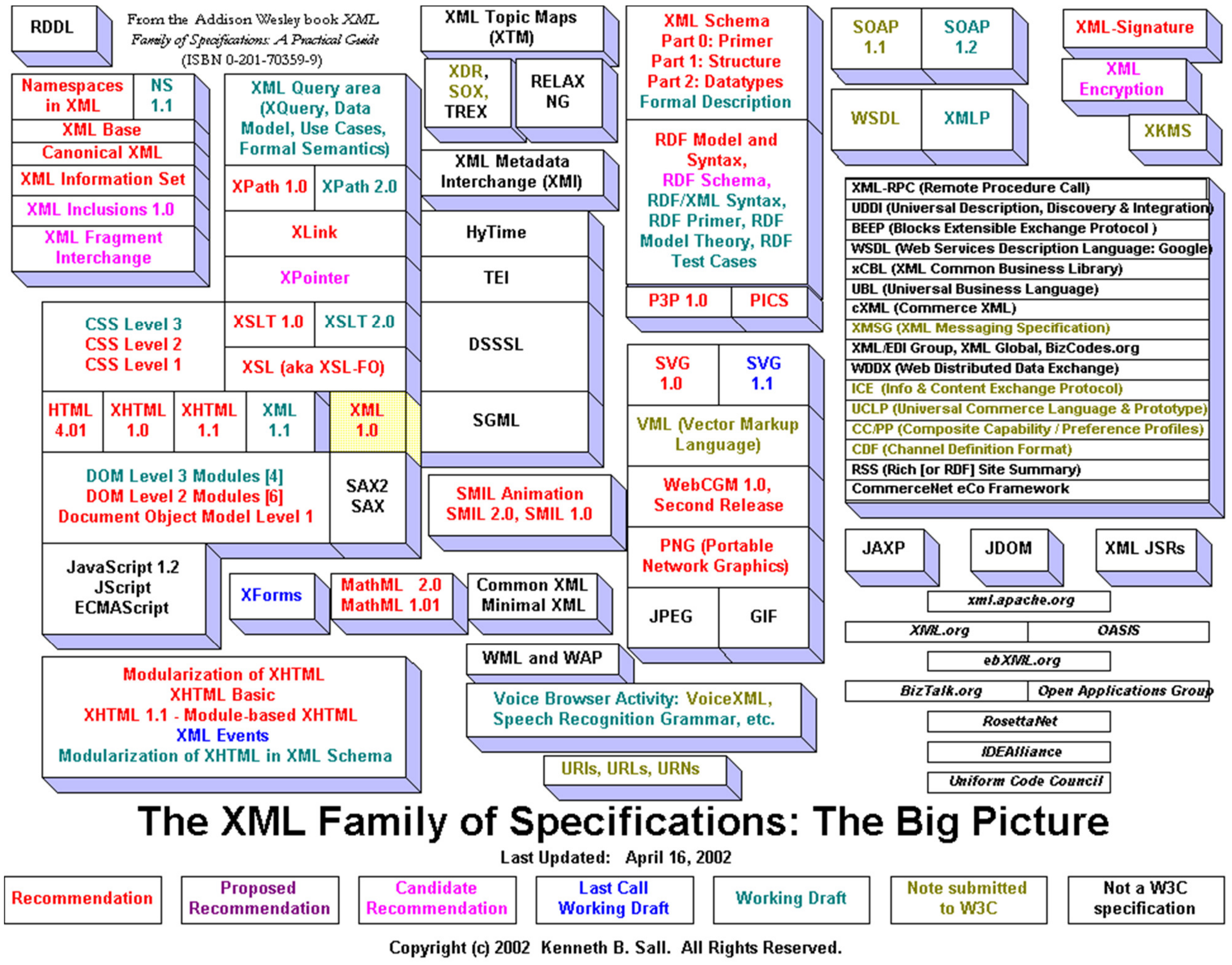
Cisco's Construction	Arista's Construction
"extensible": a property of a computer language that allows the user to add new features or modify existing ones "markup language": a computer language that allows the user to add identifiers to a document for indicating logical components or layout	markup language defined by one of the versions of the XML standard published by the W3C organization

39. I agree with Cisco's proposed construction because it comports with the intrinsic and extrinsic evidence. "Extensible markup language" is a generic term that does not denote to one of ordinary skill in the art a particular language.

40. Extensible markup languages have been used in the Internet since at least the use of the HyperText Markup Language (HTML) in the early 1990s. It is loosely based on an extension of the Standard General Markup Language (SGML) defined by the International Standards Organization (ISO) in 1986 (ISO 8879). The current HTML standard (last updated October 29, 2015, *see,* https://html.spec.whatwg.org/multipage/introduction.html, Appendix C), includes a description of its extensibility mechanisms (*see,* Section 1.7.3). At least some of these extensions date to early versions of HTML.

41. Another example of an extensible markup language is the eXtensible Markup Language (XML), also a subset of SGML and standardized by the World Wide Web Consortium (W3C) (*see, e.g.,* http://www.w3.org/TR/REC-xml/, Appendix C). XML is a currently on its fifth

revision. As its name suggests, it too is extensible. Further, XML is described as a "family of specifications" that is not a single protocol as "the" XML, but rather a meta-language for defining other languages. One representation shows, as of 2002, some of the extensible markup languages:



RDDL
From the Addison Wesley book XML Family of Specifications: A Practical Guide (ISBN 0-201-70359-9)
XML Topic Maps (XTM)
XML Schema Part 0: Primer Part 1: Structure Part 2: Datatypes Formal Description
SOAP 1.1
SOAP 1.2
XML-Signature
XML Encryption
XKMS
WSDL
XMLP
Namespaces in XML
NS 1.1
XML Base
Canonical XML
XML Information Set
XML Inclusions 1.0
XML Fragment Interchange
XML Query area (XQuery, Data Model, Use Cases, Formal Semantics)
XDR, SOX, TREX
RELAX NG
XML Metadata Interchange (XMI)
RDF Model and Syntax, RDF Schema, RDF/XML Syntax, RDF Primer, RDF Model Theory, RDF Test Cases
XPath 1.0
XPath 2.0
XLink
XPointer
HyTime
TEI
DSSSL
SGML
P3P 1.0
PICS
XML-RPC (Remote Procedure Call)
UDDI (Universal Description, Discovery & Integration)
BEEP (Blocks Extensible Exchange Protocol)
WSDL (Web Services Description Language: Google)
xCBL (XML Common Business Library)
UBL (Universal Business Language)
cXML (Commerce XML)
XMSG (XML Messaging Specification)
XML/EDI Group, XML Global, BizCodes.org
WDDX (Web Distributed Data Exchange)
ICE (Info & Content Exchange Protocol)
UCLP (Universal Commerce Language & Protocol)
CC/PP (Composite Capability / Preference Profiles)
CDF (Channel Definition Format)
RSS (Rich [or RDF] Site Summary)
CommerceNet eCo Framework
CSS Level 3 CSS Level 2 CSS Level 1
XSLT 1.0
XSLT 2.0
XSL (aka XSL-FO)
SVG 1.0
SVG 1.1
HTML 4.01
XHTML 1.0
XHTML 1.1
XML 1.1
XML 1.0
VML (Vector Markup Language)
DOM Level 3 Modules [4] DOM Level 2 Modules [6] Document Object Model Level 1
SAX2 SAX
SMIL Animation SMIL 2.0, SMIL 1.0
WebCGM 1.0, Second Release
PNG (Portable Network Graphics)
JAXP
JDOM
XML JSRs
JavaScript 1.2 JScript ECMAScript
XForms
MathML 2.0 MathML 1.01
Common XML Minimal XML
JPEG
GIF
xml.apache.org
XML.org
OASIS
ebXML.org
WML and WAP
BizTalk.org
Open Applications Group
Modularization of XHTML XHTML Basic XHTML 1.1 - Module-based XHTML XML Events Modularization of XHTML in XML Schema
Voice Browser Activity: VoiceXML, Speech Recognition Grammar, etc.
RosettaNet
IDEAlliance
URIs, URLs, URNs
Uniform Code Council
The XML Family of Specifications: The Big Picture
Last Updated: April 16, 2002
Recommendation
Proposed Recommendation
Candidate Recommendation
Last Call Working Draft
Working Draft
Note submitted to W3C
Not a W3C specification
Copyright (c) 2002 Kenneth B. Sall. All Rights Reserved.

Kenneth B. Sall, *XML Family of Specifications: A Practical Guide* (2002).

42. A demonstration of the flexibility and evolution of extensible markup languages is the standardization of XHTML, or the Extensible HyperText Markup Language. Also standardized by the W3C (*see, e.g.*, http://www.w3.org/TR/xhtml1/, Appendix C), XHTML is an extension of HTML4 that is designed to work in conjunction with W3C's XML (*see, e.g.*, http://www.w3.org/TR/xhtml1/#xhtml, Appendix C).

43. In 2003-2005, each of these extensible markup languages existed and was in use. Additional extensible markup languages, based on W3C's XML format were also in use. Another example is SOAP, originally an acronym for the Simple Object Access Protocol. SOAP is described in W3C recommendations (*see, e.g.,* http://www.w3.org/TR/2000/NOTE-SOAP-20000508/, Appendix C) and Internet Engineering Task Force (IETF) Internet Drafts. SOAP is described as an XML based protocol. Numerous other protocols that are based on W3C's XML standard can themselves be considered extensible markup languages.

44. Cisco's proposed construction is consistent with the fact that in the art, many different languages are considered to be "extensible markup languages." Cisco's proposed construction also comports with how one of ordinary skill in the art would understand the terms "extensible" and "markup language," as shown in the dictionary definitions below:

> McGraw-Hill Dictionary of Scientific and Technical Terms, 6th Ed (2003): Extensible language: a programming language which can be modified by adding new features of changing existing ones.
>
> McGraw-Hill Dictionary of Scientific and Technical Terms, 6th Ed (2003): Markup: the process of adding information (tags) to an electronic document that are not part of the content but describe its structure or elements.
>
> Dictionary Of Computer Science, Engineering, and Technology (2001): Markup language: one of any languages for annotation of source code to simply improve the source code's appearance with the means of bold-faced keywords, slanted comments, etc. In computerized document preparation, a method of adding information to the text indicating the logical components of a document, or instructions for layout of the text on the page or other information which can be interpreted by some automatic system.

45. Cisco's proposed construction is also reflected in the intrinsic evidence, where the '886 patent specification repeatedly states that the invention is not limited to one particular type of XML:

> In another embodiment, these command statements can be formatted in accordance with another set of rules and behaviors, or ***presented in a language other than XML or CLI***.

‘886 Patent at 3:26-29 (emphasis added).

> In other embodiments, a response statement could be translated in accordance with a different set of rules or behaviors, or ***outputted in languages other than XML***.

Id. at 3:50-52 (emphasis added)).

> According to one embodiment, the input was originally formatted according to an XML schema of the CLI rules and behaviors. ***In other embodiments, the input might be received in a different language*** and translated into CLI.

Id. at 4:27-30 (emphasis added).

46. Given the intrinsic and extrinsic evidence explained above, I disagree with Arista’s proposed construction. A person of ordinary skill in the art, after reading the ‘886 patent, would have no reason to believe that the invention disclosed is limited to a “markup language defined by one of the versions of the XML standard published by the W3C organization.” Just the opposite, the ‘886 patent explicitly teaches to him or her that the invention is not so limited.

47. Arista’s construction is also contrary to how one of ordinary skill would understand the term “extensible markup language (XML).” As explained above, one of ordinary skill in the art in 2003 or 2005 would understand the term to encompass many different versions and flavors of extensible markup languages, and not just a single version that is published by W3C.

2. <u>“parsing the output message to identify at least one CLI token”</u>

Cisco’s Construction	Arista’s Construction
analyzing the output message to extract at least one unit of CLI characters in a sequence	breaking down the output message into its constituent character strings and determining that at least one such string corresponds to a CLI keyword or parameter

48. I agree with Cisco’s proposed construction because it comports with the intrinsic and extrinsic evidence. Construing “CLI token” as “one unit of CLI characters in a sequence” comports with the understanding of one of ordinary skill in the art in 2003 or 2005. Technical

dictionaries at the time define "token" similar to how Cisco has construed the term:

> <u>McGraw-Hill Dictionary of Scientific and Technical Terms, 6th Ed (2003)</u>: Token: 1. A distinguishable unit in a sequence of characters.

Unlike Arista's construction, it was not commonly understood in the art that a token must correspond to a particular keyword or parameter. As it is commonly understood in the art, there is no requirement that there be a one-to-one mapping between tokens and keywords or parameters when parsing using a grammar to identify tokens. This is also reflected in the intrinsic evidence, where in an embodiment of translating CLI statements into XML statements, a single CLI token may correspond to multiple CLI keywords when multiple CLI keywords are merged for a single CLI token:

> In step 440 of flowchart 400, in one embodiment, ***each CLI token has the following rule applied to it.*** The CLI parse graph stored in memory 120 is traversed, with respect to each token. In this embodiment, the parse graph has been modified slightly from preceding implementations, to add additional information to IOS parse node types that represent CLI keywords and parameters. ***In the case of keyword nodes, the following information is added***: "parent_label," which is a label given to AND nodes; "is_boolean," which is true if the keyword is a boolean value; ***and "has_more_tag," which is true if the keyword is to be merged with the next keyword***.

'886 Patent at 6:30-40 (emphasis added).

49. Because Arista's proposed construction is inconsistent with the intrinsic and extrinsic evidence, I disagree with Arista's proposed construction. Likewise, I agree with Cisco's proposed construction because it comports with the intrinsic and extrinsic evidence.

V. <u>THE '526 PATENT</u>

50. The '526 patent, titled "Generic command interface for multiple executable routines," was issued on May 16, 2006, from an application filed on June 28, 2000.

51. The '526 patent generally relates to "command and interface control of Operating Administration and Monitoring (OAM) executable routines within software systems." '526 Patent

at 1:7-9. The ’529 patent describes that these “software systems” having “executable routines” include, for example, unified messaging systems or network management tools. *See* ’526 Patent at 1:10-27.

52. According to the ’526 patent, “system administrators may attempt to utilize multiple tools within a software system in order to increase the available administration and diagnostic tools.” ’526 Patent at 1:28-30. Thus, “there is a need [to]…integrate [] command and control functionality for multiple programs, so the user does not need to learn the command formats and syntax for each program.” *Id.* at 1:41-44. Accordingly, the invention of the ’526 patent allows a user to utilize “a simple command language…for control[ing]…multiple [such] programs having respective command formats.” *Id.* at 45-47.

53. Within this context, the ’526 patent describes the use of a parser to interpret user-entered text commands to configure or manage, for example, network equipment such as routers and/or switches. ’526 Patent at *Abstract*.

54. The ’526 patent allows a user to use a set of universal (or generic) commands. The parser interprets the user entered commands and determines (1) the correct management program to execute and (2) the command for that management program that corresponds to the user entered command.

55. More specifically, the parser uses a command parse tree whose element(s) correspond to each element of a user-entered, generic command and is able to determine a best match for the generic command entered by the user. The parser then issues a prescribed command for the management programs.

A. LEVEL OF ORDINARY SKILL IN THE ART

56. I am familiar with the knowledge of a person of ordinary skill in the field related to the subject matter of the ‘526 patent in the 2000 time period. In forming my opinion, I considered

several things, including the various approaches to input/output interfaces of routing systems employed in the art, the type of problems encountered, and the rapidity with which innovations were made. I also considered the sophistication of the technology involved, and the educational background and experience of those actively working in the field. Finally, I placed myself back in the 2000 time frame, and considered the engineers that I had taught and worked with in the networking industry. For the purposes of this declaration, I am of the opinion that a person of ordinary skill in the art with respect to the patents would have a Bachelor's of science degree in electrical engineering, computer science or engineering, or a related field, and two to four years of work or research experience in the field of networking or information networks, or a Master's degree and one to two years of experience.

57. I meet these criteria and consider myself a person with at least ordinary skill in the art pertaining to the '526 patent. I was such a person by at least 2000.

58. I understand Cisco has asserted that the claims of the '526 patent are entitled to a priority date at least as early as June 28, 200. My opinions with respect to the level of ordinary skill are also applicable to the 1999-2000 time frame.

B. <u>CLAIM TERMS OF THE 526 PATENT</u>

1. <u>"management programs"</u>

Cisco's proposed construction	Arista's proposed construction
"separate tools or external agents having their own respective command formats that provide management functions"	"tools that are configured to execute user-entered commands having their own respective command formats rather than the generic command format"

59. I agree with Cisco's construction because it is consistent with and based on the specification.

60. The '526 patent describes, and Cisco's construction recognizes, that management

programs must be separate or external agents, programs or routines:

> "FIG. 1 is a diagram of a system configured for executing a plurality of ***management programs*** according to respective command formats according to an embodiment of the present invention. The processor based system 10 includes a user input interface 12, for example a terminal interface, that enables a user to input a generic command string, described below. The processor based system 10 also includes a parser 14 configured for validating the generic command received by the user input interface 12 from the user, and translators 16 configured for issuing commands to respective ***management programs*** 18 according to respective command formats. As shown in FIG. 1, the management programs 18, implemented for example by ***different OAM tools such as RTM programs***, may be executed within the processor based system or externally as ***external agents*** accessible using a prescribed application programming interface (API). The ***management programs*** 18 may provide different administration and maintenance functions, for example initiating various real-time screens used to monitor the internal state of executable processes within the software based system 10; alternately, ***different tools*** 18 may allow the user to control the various states within the various component of the software based system 10 via ***external programs*** (e.g., programs 18 c or 18 d), or may be used to issue ***external*** alarms (e.g., SNMP manager scripts) for ***external routines*** such as message waiting indicator routines." '526 Patent at 2:57-3:15 (emphasis added).

> "As shown in FIG. 1, the management programs 18, implemented for example by ***different OAM tools such as RTM programs***, may be executed within the processor based system or externally as external agents accessible using a prescribed application programming interface (API). The management programs 18 may provide different administration and maintenance functions, for example initiating various real-time screens used to monitor the internal state of executable processes within the software based system 10; alternately, ***different tools*** 18 may allow the user to control the various states within the various component of the software based system 10 via ***external programs*** (e.g., programs 18 c or 18 d), or may be used to issue ***external*** alarms (e.g., SNMP manager scripts) for ***external routines*** such as message waiting indicator routines" '526 Patent at 3:1-15 (emphasis added).

61. As recited in the claims and described in the specification, "separate" or "external" refers to separate or external routines or programs, as distinguished from physically separate or external computers.

62. Moreover, the "separate" or "external" nature of management programs is consistent with the purpose of the invention. The specification describes that the invention of the '526 patent addresses "[a] disadvantage of utilizing many different tools[,]" namely that "that each tool 18 tends

to have its own screen and/or command, providing difficulties for the system administrator to determine which tool is the best tool (and/or which is the best syntax) to use for a given problem." '526 Patent at 3:16-20; *see also* '526 Patent at 1:28-37. This "disadvantage" is addressed by the invention of the '526 patent by "eliminating the necessity that the user needs to learn the detailed command formats and syntax." '526 Patent at *Abstract*. If management programs were not separate or external, the stated need for the invention simply would not exist as these components would be a part of the same program or routine.

2. <u>"generic command"</u>

Cisco's Construction	Arista's Construction
"command that provides an abstraction of the tool-specific command formats and syntax, enabling a user to issue the command based on the relative functions, as opposed to the specific syntax for a corresponding tool"	"command having a format and syntax that is an abstraction of the command formats and syntaxes of more than one management program, as opposed to the specific syntax for any such management program"

63. I agree with Cisco's construction because it is based on the precise description of the term in the '526 patent. The specification describes generic command as "provid[ing] a generic instruction set that provides an abstraction of the tool-specific command formats and syntax, enabling a user to issue command based on the relative functions, as opposed to the specific syntax for a corresponding tool." '526 Patent at 3:32-36.

64. In contrast, Arista's proposed construction, while similar to Cisco's construction on its face, is problematic for two reasons.

65. First, and most importantly, Arista's construction is vague and its precise meaning is difficult to ascertain. Thus, the issues I describe below are under one potential interpretation of Arista's construction. The vagueness of Arista's construction allows for other possible

interpretations.

66. Second, Arista's construction appears to import one significant limitation that is contrary to disclosures of the '526 patent. Specifically, Arista's construction requires that a generic command be an abstraction of "more than one management program."

67. The '526 patent provides explicit examples of a generic command that is an abstraction of one management program. For example, Appendix Part A shows that in one embodiment, the generic command "stop system" maps to "obs –o APP –s down." '526 Patent at Appendix Part A. Arista's construction would exclude this embodiment.

68. Under Arista's proposed construction, whether a command meets this claim limitation appears dependent on the management programs that are installed, a requirement not found anywhere in the '526 patent. For example, in the embodiment disclosed in Appendix Part A, "stop system" would not fall under Arista's construction. However, if a second management program is (1) installed and (2) the same generic command is also an abstraction of that second management program, "stop system" would now fall within the scope of Arista's construction. These additional requirements are not supported by the '526 patent.

69. Cisco's construction is also consistent with how a person of ordinary skill in the art would understand the term "abstraction." In Computer Science, abstraction is a concept that allows suppressing the specific details of how something is implemented, *e.g.*, like a black box, so the user need only know about the input and output of the black box.

70. Certainly, a generic command could be an abstraction of the commands of two or more management programs, but for the same reason, a generic command could be an abstraction of the commands of one management program, such as those disclosed in Appendix Part A of the '526 patent. Thus, Cisco's construction properly recognizes that "abstraction" is a relationship between the generic command and a tool-specific management program, and in no way connected

to the number of management programs involved, while Arista's construction improperly adds limitations that are not supported by the specification.

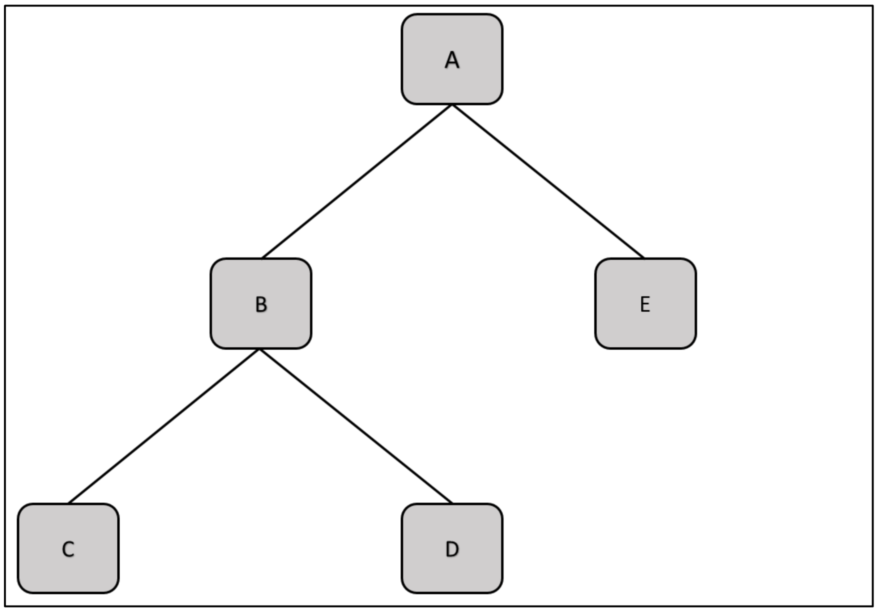
3. <u>**"recursively traversing the command parse tree based on an order of the input command words"**</u>

Cisco's Construction	Arista's Construction
Plain and ordinary meaning (except for specific terms appearing within the phrase that are otherwise terms for construction)	"recursively": "by using a function that calls itself," "traversing the command parse tree based on an order of the input command words": "sequentially determining the presence of each word of an input command in a node of a command parse tree, such that the order of the words (e.g., first, second, third) corresponds to the hierarchy of the nodes (e.g., parent, child, grandchild)."

71. I agree with Cisco that this term requires no construction by the Court. "Recursively traversing [a]…tree" is a concept that is well known in the art.

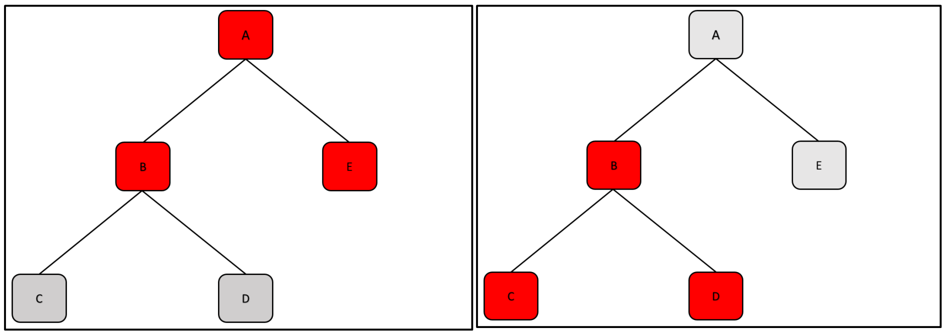
72. In contrast, Arista's construction improperly breaks down this claim term and requires a narrow, specific computer programming implementation. At the same time, Arista's construction is vague and its precise meaning is difficult to ascertain. Thus, the issues I describe below are under one potential interpretation of Arista's construction. The vagueness of their construction allows for other possible interpretations.

73. A tree is a hierarchical data structure comprised of sub-trees. Each sub-tree has the same characteristics, namely a root node with child nodes. For example, in the diagram below, elements/nodes [A] – [E] is a tree:



74. The same tree is comprised of the following sub-trees:

- [A], [B], [E] and
- [B].[C]. [D]



75. In the art, traversing a tree is a high-level concept that describes systemically processing each element of a tree. For example, the Microsoft dictionary that both parties have cited in their disclosures defines "traverse" as: "*vb*. In programming, to access in a particular order all of the nodes of a tree or similar data structure." Microsoft Computer Dictionary, Fifth Edition, c. 2002 (CSI-CLI-00403888 at 3893), Appendix E).

76. The "order" that the nodes of a tree are accessed depends on the type of traversal. For example, in the '526 patent, "recursively" modifies traversing.

77. In recursive traversals, nodes are accessed by breaking down the tree into sub-trees, as described in paragraphs 73 to 74, and further breaking down any sub-trees into their constituent sub-trees until all trees have only one node, *e.g.,* a parent node with no child nodes.

78. Cisco's construction recognizes the important distinction between recursively traversing the tree, *e.g.,* breaking down the tree into in to its constituent sub-trees and treating each sub-tree as a tree, from a recursively written computer program. The former is a path of traveling through the tree, and the latter is one of, but not the only, way to write computer code to achieve that path.

79. Arista's construction appears to require one specific implementation of recursively traversing a tree and is thus narrower than how one of ordinary skill in the art would understand the term "recursively traversing the …tree."

80. Further, Arista's construction of the word "recursion," is confined to only nested recursion, and appears to exclude at least three types of recursive functionality that are well known in the art—linear, multiple, and mutual recursion. (Manuel Rubio-Sanchez, Jamie Urquiza-Fuentes, and Cristobol Pareja-Florea, *A Gentle Introduction to Mutual Recursion,* Copyright 2008 ACM 978-1-60558-115-6/08/06 available at http://www.escet.urjc.es/~mrubio/papers/fp114-rubio-sanchez.pdf, Appendix D ("A common classification distinguishes the following types: ***linear*** (of which tail recursion is a special case), ***multiple*** (or exponential), ***nested***, and ***mutual***.")).

VI. <u>OTHER COMMENTS</u>

81. The opinions expressed in this report are my preliminary opinions based on my review to-date of the evidence produced at this stage of the case. My opinions are subject to change based on additional opinions that Cisco's experts may present and information I may receive in the future. I reserve my right to amend or update my opinions as appropriate in response to any future developments. With this in mind, based on the analysis I have conducted and for the reasons set

forth above, I have preliminarily reached the conclusions and opinions in this Declaration.

82. At a hearing or at trial I may use as exhibits various documents produced in this case that refer or relate to the matters discussed in this Declaration. I have not yet selected the particular exhibits that might be used. I may also rely on visual aids and may rely on analogies concerning elements of the patents discussed above, the accused products, the references cited in this report, or any related technologies. In addition, I may create or assist in the creation of certain demonstrative evidence to assist me in testifying, and I reserve the right to do so, such as demonstrations of devices and software to further support the positions in this Declaration.

Kevin C Almeroth

Kevin C. Almeroth

APPENDIX A

List of Materials Relied Upon

All documents and other evidence cited to in my Declaration.

U.S. Patent No. 7,047,526.

U.S. Patent No. 7,953,886.

The file history of U.S. Patent No. 7,047,526.

The file history of U.S. Patent No. 7,953,886.

Joint Claim Construction and Prehearing Statement Under Patent Local Rule 4-3 in *Cisco Systems, Inc. v. Arista Networks, Inc.*, Case No. 5:14-cv-05344-BLF, including accompanying exhibits and extrinsic evidence.

Kenneth B. Sall, *XML Family of Specifications: A Practical Guide* (2002).

https://html.spec.whatwg.org/multipage/introduction.html

http://www.w3.org/TR/REC-xml/

http://www.w3.org/TR/xhtml1/

http://www.w3.org/TR/xhtml1/#xhtml

http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

Manuel Rubio-Sanchez, Jamie Urquiza-Fuentes, and Cristobol Pareja-Florea, *A Gentle Introduction to Mutual Recursion,* Copyright 2008 ACM 978-1-60558-115-6/08/06, available at http://www.escet.urjc.es/~mrubio/papers/fp114-rubio-sanchez.pdf

APPENDIX B

Kevin C. Almeroth

Professor, Department of Computer Science
University of California
Santa Barbara, CA 93106-5110
(805)636-1123 (office)
(805)893-8553 (fax)
almeroth@cs.ucsb.edu (email)
http://www.cs.ucsb.edu/~almeroth (WWW URL)

Education

Ph.D.	June 1997	*Georgia Institute of Technology* Computer Science *Dissertation Title*: Networking and System Support for the Efficient, Scalable Delivery of Services in Interactive Multimedia Systems *Minor*: Telecommunications Public Policy
M.S.	June 1994	*Georgia Institute of Technology* Computer Science *Specialization*: Networking and Systems
B.S. **(high honors)**	June 1992	*Georgia Institute of Technology* Information and Computer Science *Minors*: Economics, Technical Communication, American Literature

Employment History

Professor	Department of Computer Science University of California Santa Barbara, CA	Jul 2005 -- present
Associate Dean	College of Engineering University of California Santa Barbara, CA	Mar 2007 -- Aug 2009
Vice Chair	Department of Computer Science University of California Santa Barbara, CA	Jul 2000 -- Nov 2005
Associate Professor	Department of Computer Science University of California	Jul 2001 -- Jun 2005

	Santa Barbara, CA	
Assistant Professor	Department of Computer Science University of California Santa Barbara, CA	Jul 1997 -- Jun 2001
Graduate Researcher	Broadband Telecommunications Center Georgia Center for Adv Telecom Tech Atlanta, GA	Sep 1996--Jun 1997
Graduate Intern	IBM T.J. Watson Research Labs Hawthorne, NY	Jun 1995--Sep 1995
Support Specialist	Office of Information Technology Georgia Institute of Technology Atlanta, GA	Sep 1995--Jun 1997
Research Assistant	College of Computing Georgia Institute of Technology Atlanta, GA	Jan 1994--Mar 1994
Graduate Intern	Hitachi Telecommunications Norcross, GA	Jun 1992--Sep 1992

Industry Technical Advising

Board of Directors	The New Media Studio Santa Barbara, CA	Nov 2006 -- present
Co-Founder & Chairman of the Board	Santa Barbara Labs, LLC Santa Barbara, CA	Sep 2007 -- Dec 2009
Board of Advisors	Techknowledge Point Santa Barbara, CA	May 2001 -- Dec 2007
Technical Advisory Board	Occam Networks, Inc. Santa Barbara, CA	May 2000 -- Dec 2010
Board of Advisors	Airplay Inc. San Francisco, CA	Jun 2005 -- Aug 2009
Consultant	Lockheed Martin Corporation San Jose, CA	Nov 1999 -- Jun 2009
Board of Advisors	Santa Barbara Technology Group Santa Barbara, CA	Sep 2000 -- Dec 2004

Board of Directors	Virtual Bandwidth, Inc. Santa Barbara, CA	Nov 2000 -- Jun 2001
Board of Advisors & Affiliated Scientist	Digital Fountain San Francisco, CA	Jan 2000 -- Dec 2001
Senior Technologist	IP Multicast Initiative, Stardust Forums Campbell, CA	Jun 1998 -- Dec 2000

I. Teaching

A. Courses Taught

CS 176A	Intro to Computer Communication Networks	Fall 1997, Fall 1998, Fall 2002, Fall 2003, Fall 2004, Spring 2005, Spring 2006, Spring 2007, Spring 2008, Fall 2008, Fall 2009, Fall 2010, Fall 2011, Fall 2012, Fall 2013, Fall 2014
CS 176B	Network Computing	Winter 2000, Winter 2001, Winter 2002, Winter 2012, Winter 2014, Winter 2015
MAT 201B	Media Networks and Services	Fall 1999, Fall 2000, Fall 2001, Fall 2003
CS 276	Distributed Computing and Computer Networks	Winter 1999, Spring 2000, Fall 2002, Fall 2005
CS 290I	Networking for Multimedia Systems	Winter 1998, Spring 1999, Fall 2004, Winter 2010
CS 595N	Technology and Society	Winter 2005, Fall 2005, Spring 2006, Fall 2006, Spring 2007, Fall 2007, Spring 2008, Fall 2008, Spring 2009
CS 595N	Economic Systems Seminar	Winter 2004, Spring 2004, Winter 2005, Spring 2005
CS 595N	Networking Seminar	Winter 1999, Fall 1999, Winter 2003
CS 595N	Wireless Networking & Multimedia Seminar	Fall 2000
CS 595I	Systems Design and Implementation Seminar	Fall 1999, Fall 2000, Winter 2001, Spring 2001, Winter 2002, Spring 2002

B. Other Teaching Experience

- *The Evolution of Advanced Networking Services: From the ARPAnet to Internet2*, Instructor, Summer 2001. Short course taught at Escuela de Ciencias Informatica (ECI) sponsored by the Universidad de Buenos Aires.

- *Johns Hopkins Center for Talented Youth*, Instructor, Summer 1994. CTY is a program to teach gifted high school students the fundamentals of computer science.
- *Georgia Institute of Technology*, Graduate Teaching Assistant, Sep 1994--Sep 1996. Worked as a TA for 12 quarters teaching 7 different courses (4 undergraduate and 3 graduate).

C. Ph.D. Students Advised [14 graduated]

14. Daniel Havey
    Research Area: *Throughput and Delay on the Packet Switched Internet*
    Date Graduated: Winter 2015
    First Position: Microsoft
13. Lara Deek (co-advised with E. Belding)
    Research Area: *Resource-Efficient Wireless Systems for Emerging Wireless Networks*
    Date Graduated: Summer 2014
    First Position: Post Doc, UIUC
12. Mike Wittie
    Research Area: *Towards Sustained Scalability of Communication Networks*
    Date Graduated: Summer 2011
    First Position: Assistant Professor, Montana State University
11. Allan Knight
    Research Area: *Supporting Integration of Educational Technologies and Research of Their Effects on Learning*
    Date Graduated: Summer 2009
    First Position: Research Scientist, Citrix Online
10. Hangjin Zhang
    Research Area: *Towards Blended Learning: Educational Technology to Improve and Assess Teaching and Learning*
    Date Graduated: Spring 2009
    First Position: Microsoft
9. Gayatri Swamynathan
    Dissertation Title: *Towards Reliable Reputations for Distributed Applications*
    Date Graduated: Spring 2008
    First Position: Zynga
8. Amit Jardosh (co-advised with E. Belding)
    Dissertation Title: *Adaptive Large-Scale Wireles Networks: Measurements, Protocol Designs, and Simulation Studies*
    Date Graduated: Fall 2007
    First Position: Yahoo!
7. Khaled Harras
    Dissertation Title: *Protocol and Architectural Challenges in Delay and Disruption Tolerant Networks*
    Date Graduated: Summer 2007
    First Position: Assistant Professor, Carnegie Mellon University
6. Krishna Ramachandran (co-advised with E. Belding)
    Dissertation Title: *Design, Deployment, and Management of High-Capacity Wireless Mesh*

Networks
Date Graduated: Winter 2006
First Position: Research Scientist, Citrix Online

5. Robert Chalmers
Dissertation Title: *Improving Device Mobility with Intelligence at the Network Edge*
Date Graduated: Summer 2004
First Position: President and CEO, Limbo.net

4. Prashant Rajvaidya
Dissertation Title: *Achieving Robust and Secure Deployment of Multicast*
Date Graduated: Spring 2004
First Position: President and CTO, Mosaic Networking

3. Sami Rollins
Dissertation Title: *Overcoming Resource Constraints to Enable Content Exchange Applications in Next-Generation Environments*
Date Graduated: Spring 2003
First Position: Assistant Professor, Mount Holyoke College

2. Srinivasan Jagannathan
Dissertation Title: *Multicast Tree-Based Congestion Control and Topology Management*
Date Graduated: Spring 2003
First Position: Consultant, Kelly & Associates

1. Kamil Sarac
Dissertation Title: *Supporting a Robust Multicast Service in the Global Infrastructure*
Date Graduated: Spring 2002
First Position: Assistant Professor, UT-Dallas

D. M.S. Students Advised (Thesis/Project Option) [19 graduated and 1 current]

20. Greg Parsons
Research Area: *Drone-Based Mesh Networks*
Date Started: Fall 2014

19. Neer Shey
Research Area: *Analyzing Content Distribution Through Opportunistic Contact for Smart Cellular Phones*
Date Graduated: Spring 2010

18. Camilla Fiorese
Research Area: *Analysis of a Pure Rate-Based Congestion Control Algorithm*
Date Graduated: Summer 2009

17. Brian Weiner
Research Area: *Multi-Socket TCP: A Simple Approach to Improve Performance of Real-Time Applications over TCP*
Date Graduated: Fall 2007

16. Avijit Sen Mazumder
Research Area: *Facilitating Robust Multicast Group Management*
Date Graduated: Fall 2005

15. Rishi Matthew
Thesis Title: *Providing Seamless Access to Multimedia Content on Heterogeneous Platforms*

Date Graduated: Summer 2004

14. Camden Ho
    Research Area: *Tools and Techniques for Wireless Network Management*
    Date Graduated: Spring 2004
13. Amit Jardosh (co-advised with E. Belding)
    Research Area: *Realistic Environment Models for Mobile Network Evaluation*
    Date Graduated: Spring 2004
12. Nitin Solanki
    Research Area: *SongWand: A Wireless Barcode Scanner Using Bluetooth Technology*
    Date Graduated: Winter 2004
11. Vrishali Wagle (co-advised with E. Belding)
    Research Area: *An Ontology-Based Service Discovery Mechanism*
    Date Graduated: Winter 2004
10. Uday Mohan
    Thesis Title: *Scalable Service Discovery in Mobile Ad hoc Networks*
    Date Graduated: Spring 2003
9. Krishna Ramachandran
    Thesis Title: *Ubiquitous Multicast*
    Date Graduated: Spring 2003
8. John Slonaker
    Thesis Title: *Inductive Loop Signature Acquisition Techniques*
    Date Graduated: Spring 2002
7. Mohammad Battah
    Thesis Title: *Dedicated Short-Range Communications Intelligent Transportation Systems Protocol (DSRC-ITS)*
    Date Graduated: Spring 2002
6. Kevin Vogel
    Thesis Title: *Integrating E-Commerce Applications into Existing Business Infrastructures*
    Date Graduated: Spring 2001
5. Sami Rollins
    Thesis Title: *Audio XmL: Aural Interaction with XML Documents*
    Date Graduated: Winter 2000
4. Andy Davis
    Thesis Title: *Stream Scheduling for Data Servers in a Scalable Interactive TV System*
    Date Graduated: Spring 1999
3. David Makofske
    Thesis Title: *MHealth: A Real-Time Graphical Multicast Monitoring Tool*
    Date Graduated: Winter 1999
2. Prashant Rajvaidya
    Thesis Title: *MANTRA: Router-Based Monitoring and Analysis of Multicast Traffic*
    Date Graduated: Winter 1999
1. Alex DeCastro (co-advised with Yuan-Fang Wang)
    Thesis Title: *Web-Based Collaborative 3D Modeling*
    Date Graduated: Winter 1998

E. Teaching Awards

2006-2007 UCSB Academic Senate Distinguished Teaching Award
2004-2005 Computer Science Outstanding Faculty Member
2000-2001 UCSB Spotlight on Excellence Award
1999-2000 Computer Science Outstanding Faculty Member (co-recipient)
1998-1999 Computer Science Outstanding Faculty Member (co-recipient)
1997-1998 Computer Science Outstanding Faculty Member

II. Research

A. Journal Papers, Magazine Articles, Books, and Book Chapters

62. L. Deek, E. Garcia-Villegas, E. Belding, S.J. Lee, and K. Almeroth, "A Practical Framework for 802.11 MIMO Rate Adaptation," Computer Networks, vol. 83, num. 6, pp. 332-348, June 2015.

61. L. Deek, E. Garcia-Villegas, E. Belding, S.J. Lee, and K. Almeroth, "Intelligent Channel Bonding in 802.11n WLANs," IEEE Transactions on Mobile Computing, vol. 13, num. 6, pp. 1242-1255, June 2014.

60. H. Zhang and K. Almeroth, "Alternatives for Monitoring and Limiting Network Access to Students in Network-Connected Classrooms," Journal of Interactive Learning Research (JILR), vol. 24, num. 3, pp. 237-265, July 2013.

59. M. Tavakolifard and K. Almeroth, "A Taxonomy to Express Open Challenges in Trust and Reputation Systems," Journal of Communications, vol. 7, num. 7, pp. 538-551, July 2012.

58. M. Tavakolifard and K. Almeroth, "Social Computing: An Intersection of Recommender Systems, Trust/Reputation Systems, and Social Networks," IEEE Network, vol. 26, num. 4, pp. 53-58, July/August 2012.

57. M. Tavakolifard, K. Almeroth, and P. Ozturk, "Subjectivity Handling of Ratings for Trust and Reputation Systems: An Abductive Reasoning Approach," International Journal of Digital Content Technology and its Applications (JDCTA), vol. 5, num. 11, pp. 359-377, November 2011.

56. R. Raghavendra, P. Acharya, E. Belding and K. Almeroth, "MeshMon: A Multi-Tiered Framework for Wireless Mesh Network Monitoring," Wireless Communications and Mobile Computing (WCMC) Journal, vol. 11, num. 8, pp. 1182-1196, August 2011.

55. A. Knight and K. Almeroth, "Automatic Plagiarism Detection with PAIRwise 2.0," Journal of Interactive Learning Research (JILR), vol. 22, num. 3, pp. 379-400, July 2011.

54. V. Kone, M. Zheleva, M. Wittie, B. Zhao, E. Belding, H. Zheng, and K. Almeroth, "AirLab: Consistency, Fidelity and Privacy in Wireless Measurements," ACM Computer Communications Review, vol. 41, num. 1, pp. 60-65, January 2011.

53. G. Swamynathan, K. Almeroth, and B. Zhao, "The Design of a Reliable Reputation System,"

Electronic Commerce Research Journal, vol. 10, num. 3-4, pp. 239-270, December 2010.

52. P. Acharya, A. Sharma, E. Belding, K. Almeroth and K. Papagiannaki, "Rate Adaptation in Congested Wireless Networks through Real-Time Measurements," IEEE Transactions on Mobile Computing, vol. 9, num. 11, pp. 1535-1550, November 2010.

51. R. Raghavendra, E. Belding, K. Papagiannaki, and K. Almeroth, "Unwanted Link Layer Traffic in Large IEEE 802.11 Wireless Networks," IEEE Transactions on Mobile Computing, vol. 9, num. 9, pp. 1212-1225, September 2010.

50. H. Zhang and K. Almeroth, "Moodog: Tracking Student Activity in Online Course Management Systems," Journal of Interactive Learning Research (JILR), vol. 21, num. 3, pp. 407-429, July 2010.

49. R. Chertov and K. Almeroth, "Qualitative Comparison of Link Shaping Techniques," International Journal of Communication Networks and Distributed Systems, vol. 5, num. 1/2, pp. 109-129, July 2010.

48. A. Knight and K. Almeroth, "Fast Caption Alignment for Automatic Indexing of Audio," International Journal of Multimedia Data Engineering and Management, vol. 1, num. 2, pp. 1-17, April-June 2010.

47. K. Harras and K. Almeroth, "Scheduling Messengers in Disconnected Clustered Mobile Networks," Ad Hoc & Sensor Wireless Networks, vol. 9, num. 3-4, pp. 275-304, March-April 2010.

46. A. Jardosh, K. Papagiannaki, E. Belding, K. Almeroth, G. Iannaccone, and B. Vinnakota, "Green WLANs: On-Demand WLAN Infrastructures," ACM Journal on Mobile Networks and Applications (MONET), vol. 14, num. 6, pp. 798-814, December 2009.

45. M. Wittie, K. Harras, K. Almeroth, and E. Belding, "On the Implications of Routing Metric Staleness in Delay Tolerant Networks," Computer Communications Special Issue on Delay and Disruption Tolerant Networking, vol. 32, num. 16, pp. 1699-1709, October 2009.

44. K. Harras, L. Deek, C. Holman, and K. Almeroth, "DBS-IC: An Adaptive Data Bundling System for Intermittent Connectivity," Computer Communications Special Issue on Delay and Disruption Tolerant Networking, vol. 32, num. 16, pp. 1687-1698, October 2009.

43. S. Karpinski, E. Belding, K. Almeroth, and J. Gilbert, "Linear Representations of Network Traffic," ACM Journal on Mobile Networks and Applications (MONET), vol. 14, num. 4, pp. 368-386, August 2009.

42. K. Harras and K. Almeroth, "Controlled Flooding in Disconnected Sparse Mobile Networks," Wireless Communications and Mobile Computing (WCMC) Journal, vol. 9, num. 1, pp. 21-33, January 2009.

41. R. Mayer, A. Stull, K. DeLeeuw, K. Almeroth, B. Bimber, D. Chun, M. Bulger, J. Campbell, A. Knight, and H. Zhang, "Clickers in College Classrooms: Fostering Learning with Questioning Methods in Large Lecture Classes," Contemporary Educational Psychology, vol. 34, num. 1, pp. 51-57, January 2009.

40. A. Knight, K. Almeroth, and B. Bimber, "Design, Implementation and Deployment of PAIRwise," Journal of Interactive Learning Research (JILR), vol. 19, num. 3, pp. 489-508, July 2008.

39. A. Garyfalos and K. Almeroth, "Coupons: A Multilevel Incentive Scheme for Information Dissemination in Mobile Networks," IEEE Transactions on Mobile Computing, vol. 7, num. 6, pp. 792-804, June 2008.

38. I. Sheriff, K. Ramachandran, E. Belding, and K. Almeroth, "A Multi-Radio 802.11 Mesh Network Architecture," ACM Journal on Mobile Networks and Applications (MONET), vol. 13, num. 1-2, pp. 132-146, April 2008.

37. M. Bulger, R. Mayer, K. Almeroth, and S. Blau, "Measuring Learner Engagement in Computer-Equipped College Classrooms," Journal of Educational Multimedia and Hypermedia, vol. 17, num. 2, pp. 129-143, April 2008.

36. G. Swamynathan, B. Zhao, and K. Almeroth, "Exploring the Feasibility of Proactive Reputations," Concurrency and Computation: Practice and Experience, vol. 20, num. 2, pp. 155-166, February 2008.

35. G. Swamynathan, B. Zhao, K. Almeroth, and H. Zheng, "Globally Decoupled Reputations for Large Distributed Networks," Advances in Multimedia, vol. 2007, pp. 1-14, 2007.

34. R. Mayer, A. Stull, J. Campbell, K. Almeroth, B. Bimber, D. Chun and A. Knight, "Overestimation Bias in Self-reported SAT Scores," Educational Psychology Review, vol. 19, num. 4, pp. 443-454, December 2007.

33. P. Namburi, K. Sarac and K. Almeroth, "Practical Utilities for Monitoring Multicast Service Availability," Computer Communications Special Issue on Monitoring and Measurement of IP Networks, vol. 29, num. 10, pp. 1675-1686, June 2006.

32. R. Chalmers, G. Krishnamurthi and K. Almeroth, "Enabling Intelligent Handovers in Heterogeneous Wireless Networks," ACM Journal on Mobile Networks and Applications (MONET), vol. 11, num. 2, pp. 215-227, April 2006.

31. H. Lundgren, K. Ramachandran, E. Belding, K. Almeroth, M. Benny, A. Hewatt, A. Touma and A. Jardosh, "Experience from the Design, Deployment and Usage of the UCSB MeshNet Testbed," IEEE Wireless Communications, vol. 13, num. 2, pp. 18-29, April 2006.

30. R. Mayer, K. Almeroth, B. Bimber, D. Chun, A. Knight and A. Campbell, "Technology Comes to College: Understanding the Cognitive Consequences of Infusing Technology in College Classrooms," Educational Technology, vol. 46, num. 2, pp. 48-53, March-April 2006.

29. A. Garyfalos and K. Almeroth, "A Flexible Overlay Architecture for Mobile IPv6 Multicast," Journal on Selected Areas in Communications (JSAC) Special Issue on Wireless Overlay Networks Based on Mobile IPv6, vol. 23, num. 11, pp. 2194-2205, November 2005.

28. K. Sarac and K. Almeroth, "Monitoring IP Multicast in the Internet: Recent Advances and Ongoing Challenges," IEEE Communications, vol. 43, num. 10, pp. 85-91, October 2005.

27. K. Sarac and K. Almeroth, "Application Layer Reachability Monitoring for IP Multicast," Computer Networks, vol. 48, num. 2, pp. 195-213, June 2005.

26. A. Jardosh, E. Belding, K. Almeroth and S. Suri, "Real-world Environment Models for Mobile Network Evaluation," Journal on Selected Areas in Communications Special Issue on Wireless Ad hoc

Networks, vol. 23, num. 3, pp. 622-632, March 2005.

25. S. Rollins and K. Almeroth, "Evaluating Performance Tradeoffs in a One-to-Many Peer Content Distribution Architecture," Journal of Internet Technology, vol. 5, num. 4, pp. 373-387, Fall 2004.

24. K. Sarac and K. Almeroth, "Tracetree: A Scalable Mechanism to Discover Multicast Tree Topologies in the Network," IEEE/ACM Transactions on Networking, vol. 12, num. 5, pp. 795-808, October 2004.

23. K. Sarac and K. Almeroth, "A Distributed Approach for Monitoring Multicast Service Availability," Journal of Network and Systems Management, vol. 12, num. 3, pp. 327-348, September 2004.

22. P. Rajvaidya, K. Ramachandran and K. Almeroth, "Managing and Securing the Global Multicast Infrastructure," Journal of Network and Systems Management, vol. 12, num. 3, pp. 297-326, September 2004.

21. P. Rajvaidya and K. Almeroth, "Multicast Routing Instabilities," IEEE Internet Computing, vol. 8, num. 5, pp. 42-49, September/October 2004.

20. D. Johnson, R. Patton, B. Bimber, K. Almeroth and G. Michaels, "Technology and Plagiarism in the University: Brief Report of a Trial in Detecting Cheating," Association for the Advancement of Computing in Education (AACE) Journal, vol. 12, num. 3, pp. 281-299, Summer 2004.

19. R. Chalmers and K. Almeroth, "A Security Architecture for Mobility-Related Services," Journal of Wireless Personal Communications, vol 29, num. 3, pp. 247-261, June 2004.

18. B. Stiller, K. Almeroth, J. Altmann, L. McKnight, and M. Ott, "Pricing for Content in the Internet," Computer Communications, vol. 27, num. 6, pp. 522-528, April 2004.

17. S. Rollins and K. Almeroth, "Lessons Learned Deploying a Digital Classroom," Journal of Interactive Learning Research (JILR), vol. 15, num. 2, pp. 169-185, April 2004.

16. S. Jagannathan and K. Almeroth, "A Dynamic Pricing Scheme for E-Content at Multiple Levels-of-Service," Computer Communications, vol. 27, num. 4, pp. 374-385, March 2004.

15. K. Almeroth, "Using Satellite Links in the Delivery of Terrestrial Multicast Traffic," Internetworking and Computing over Satellites, Kluwer Academic Publishers, 2003.

14. R. Chalmers and K. Almeroth, "On the Topology of Multicast Trees," IEEE/ACM Transactions on Networking, vol. 11, num. 1, pp. 153-165, January 2003.

13. S. Jagannathan, J. Nayak, K. Almeroth, and M. Hofmann, "On Pricing Algorithms for Batched Content Delivery Systems," Electronic Commerce Research and Applications Journal, vol. 1, num. 3-4, pp. 264-280, Fall 2002.

12. D. Makofske and K. Almeroth, "Multicast Sockets: Practical Guide for Programmers," *Morgan Kaufmann Publishers*, November 2002.

11. S. Jagannathan and K. Almeroth, "Price Issues in Delivering E-Content On-Demand," ACM Sigecom Exchanges, vol. 3, num. 2, pp. 18-27, May 2002.

10. D. Makofske and K. Almeroth, "From Television to Internet Video-on-Demand: Techniques and Tools

for VCR-Style Interactivity," Software: Practice and Experience, vol. 31, num. 8, pp. 781-801, July 2001.

9. K. Sarac and K. Almeroth, "Supporting Multicast Deployment Efforts: A Survey of Tools for Multicast Monitoring," Journal on High Speed Networking, Special Issue on Management of Multimedia Networking, vol. 9, num. 3/4, pp. 191-211, March 2001.

8. K. Almeroth, "Adaptive, Workload-Dependent Scheduling for Large-Scale Content Delivery Systems," Transactions on Circuits and Systems for Video Technology, *Special Issue on Streaming Video*, vol. 11, num. 3, pp. 426-439, March 2001.

7. D. Makofske and K. Almeroth, "Real-Time Multicast Tree Visualization and Monitoring," Software: Practice and Experience, vol. 30, num. 9, pp. 1047-1065, July 2000.

6. M. Ammar, K. Almeroth, R. Clark and Z. Fei, "Multicast Delivery of WWW Pages," Electronic Commerce Technology Trends: Challenges and Opportunities, IBM Press, February 2000.

5. K. Almeroth, "The Evolution of Multicast: From the MBone to Inter-Domain Multicast to Internet2 Deployment," IEEE Network Special Issue on Multicasting, vol. 10, num. 1, pp. 10-20, January/February 2000.

4. K. Almeroth and M. Ammar, "An Alternative Paradigm for Scalable On-Demand Applications: Evaluating and Deploying the Interactive Multimedia Jukebox," IEEE Transactions on Knowledge and Data Engineering Special Issue on Web Technologies, vol. 11, num. 4, pp 658-672, July/August 1999.

3. K. Almeroth and M. Ammar, "The Interactive Multimedia Jukebox (IMJ): A New Paradigm for the On-Demand Delivery of Audio/Video," Computer Networks and ISDN Systems, vol. 30, no. 1, April 1998.

2. K. Almeroth and M. Ammar, "Multicast Group Behavior in the Internet's Multicast Backbone (MBone)," IEEE Communications, vol. 35, no. 6, pp. 124-129, June 1997.

1. K. Almeroth and M. Ammar, "On the Use of Multicast Delivery to Provide a Scalable and Interactive Video-on-Demand Service," Journal on Selected Areas of Communication (JSAC), vol. 14, no. 6, pp. 1110-1122, August 1996.

B. Conference Papers with Proceedings (refereed)

89. D. Havey and K. Almeroth, "Active Sense Queue Management (ASQM)," *IFIP Networking Conference*, Toulouse, FRANCE, May 2015.

88. L. Deek, E. Garcia-Villegas, E. Belding, S.J. Lee, and K. Almeroth, "Joint Rate and Channel Width Adaptation in 802.11 MIMO Wireless Networks," IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), New Orleans, LA, USA, June 2013.

87. D. Havey and K. Almeroth, "Fast Wireless Protocol: A Network Stack Design for Wireless Transmission," *IFIP Networking Conference*, Brooklyn, New York, USA, May 2013.

86. M. Tavakolifard, J. Gulla, K. Almeroth, J. Ingvaldsen, G. Nygreen, and E. Berg, "Tailored News in the Palm of Your HAND: A Multi-Perspective Transparent Approach to News Recommendation," *Demo Track at the International World Wide Web Conference (WWW)*, Rio de Janeiro, BRAZIL, May 2013.

85. S. Patterson, M. Wittie, K. Almeroth, and B. Bamieh, "Network Optimization with Dynamic Demands and Link Prices," *Allerton Conference*, Monticello, Illinois, USA, October 2012.

84. D. Havey, R. Chertov, and K. Almeroth, "Receiver Driven Rate Adaptation," *ACM Multimedia Systems Conference (MMSys)*, Chapel Hill, North Carolina, USA, February 2012.

83. M. Tavakolifard and K. Almeroth, "Trust 2.0: Who to Believe in the Flood of Online Data?" *International Conference on Computing, Networking and Communications (ICNC)*, Maui, Hawaii, USA, January 2012.

82. L. Deek, E. Garcia-Villegas, E. Belding, S.J. Lee, and K. Almeroth, "The Impact of Channel Bonding on 802.11n Network Management," *ACM CoNEXT*, Tokyo, JAPAN, December 2011.

81. L. Deek, X. Zhou, K. Almeroth, and H. Zheng, "To Preempt or Not: Tackling Bid and Time-based Cheating in Online Spectrum Auctions," *IEEE Infocom*, Shanghai, CHINA, April 2011.

80. M. Wittie, V. Pejovic, L. Deek, K. Almeroth, and B. Zhao, "Exploiting Locality of Interest in Online Social Networks," *ACM CoNEXT*, Philadelphia, Pennsylvania, USA, November 2010.

79. R. Chertov and K. Almeroth, "Using BGP in a Satellite-Based Challenged Network Environment," *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, Boston, Massachusetts, USA, June 2010.

78. R. Chertov, D. Havey and K. Almeroth, "MSET: A Mobility Satellite Emulation Testbed," *IEEE Infocom*, San Diego, California, USA, March 2010.

77. B. Stone-Gross, A. Moser, C. Kruegel, E. Kirda, and K. Almeroth, "FIRE: FInding Rogue nEtworks," *Annual Computer Security Applications Conference (ACSAC)*, Honolulu, Hawaii, USA, December 2009.

76. M. Wittie, K. Almeroth, E. Belding, I. Rimac, and V. Hilt, "Internet Service in Developing Regions Through Network Coding," *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, Rome, ITALY, June 2009.

75. R. Chertov and K. Almeroth, "High-Fidelity Link Shaping," *International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM)*, Washington DC, USA, April 2009.

74. L. Deek, K. Almeroth, M. Wittie, and K. Harras, "Exploiting Parallel Networks Using Dynamic Channel Scheduling," *International Wireless Internet Conference (WICON)*, Maui, Hawaii, USA, November 2008.

73. D. Havey, E. Barlas, R. Chertov, K. Almeroth, and E. Belding, "A Satellite Mobility Model for QUALNET Network Simulations," *IEEE Military Communications Conference (MILCOM)*, San Diego, California, USA, November 2008.

72. J. Kayfetz and K. Almeroth, "Creating Innovative Writing Instruction for Computer Science Graduate Students," *ASEE/IEEE Frontiers in Education (FIE) Conference*, Saratoga Springs, New York, USA, October 2008.

71. G. Swamynathan, B. Zhao, K. Almeroth, and S. Rao, "Towards Reliable Reputations for Dynamic Networked Systems," *IEEE International Symposium on Reliable Distributed Systems (SRDS)*, Napoli, ITALY, October 2008.

70. B. Stone-Gross, D. Sigal, R. Cohn, J. Morse, K. Almeroth, and C. Krugel, "VeriKey: A Dynamic Certificate Verification System for Public Key Exchanges," *Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, Paris, FRANCE, July 2008.

69. P. Acharya, A. Sharma, E. Belding, K. Almeroth, K. Papagiannaki, "Congestion-Aware Rate Adaptation in Wireless Networks: A Measurement-Driven Approach," *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, San Francisco, California, USA, June 2008.

68. A. Jardosh, P. Suwannatat, T. Hollerer, E. Belding, and K. Almeroth, "SCUBA: Focus and Context for Real-time Mesh Network Health Diagnosis," *Passive and Active Measurement Conference (PAM)*, Cleveland, Ohio, USA, April 2008.

67. B. Stone-Gross, C. Wilson, K. Almeroth, E. Belding, H. Zheng, K. Papagiannaki, "Malware in IEEE 802.11 Wireless Networks," *Passive and Active Measurement Conference (PAM)*, Cleveland, Ohio, USA, April 2008.

66. R. Raghavendra, E. Belding, K. Papagiannaki, and K. Almeroth, "Understanding Handoffs in Large IEEE 802.11 Wireless Networks," *Internet Measurement Conference (IMC)*, San Diego, California, USA, October 2007.

65. M. Wittie, B. Stone-Gross, K. Almeroth and E. Belding, "MIST: Cellular Data Network Measurement for Mobile Applications," *IEEE International Conference on Broadband Communications, Networks, and Systems (BroadNets)*, Raleigh, North Carolina, USA, September 2007.

64. S. Karpinski, E. Belding, K. Almeroth, "Wireless Traffic: The Failure of CBR Modeling," *IEEE International Conference on Broadband Communications, Networks, and Systems (BroadNets)*, Raleigh, North Carolina, USA, September 2007.

63. A. Knight, K. Almeroth, H. Zhang, R. Mayer, and K. DeLeeuw, "Data Cafe: A Dining Car Approach to Educational Research Data Management and Distribution," *World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED MEDIA)*, Vancouver, CANADA, June 2007.

62. H. Zhang, K. Almeroth, A. Knight, M. Bulger, and R. Mayer, "Moodog: Tracking Students' Online Learning Activities," *World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED MEDIA)*, Vancouver, CANADA, June 2007.

61. M. Bulger, K. Almeroth, R. Mayer, D. Chun, A. Knight, H. Collins, "Effects of Instructor Engagement on Student Use of a Course Management System," Assocation for Psychological Science (APS) Annual Conference, Washington DC, USA, May 2007.

60. R. Mayer, A. Stull, K. Almeroth, B. Bimber, D. Chun, M. Bulger, J. Campbell, Allan Knight, and H.

Zhang, "Using Technology-Based Methods to Foster Learning in Large Lecture Classes: Evidence for the Pedagogic Value of Clickers," *American Educational Research Association (AERA) Annual Conference*, Chicago, Illinois, USA, April 2007.

59. K. Ramachandran, I. Sheriff, E. Belding, and K. Almeroth, "Routing Stability in Static Wireless Mesh Networks," *Passive and Active Measurement Conference (PAM)*, Louvain-la-neuve, BELGIUM, April 2007.

58. G. Swamynathan, T. Close, S. Banerjee, R. McGeer, B. Zhao, and K. Almeroth, "Scalable Access Control For Web Services," *International Conference on Creating, Connecting and Collaborating through Computing (C5)*, Kyoto, JAPAN, January 2007.

57. A. Knight, M. Bulger, K. Almeroth, and H. Zhang, "Is Learning Really a Phone Call Away? Knowledge Transfer in Mobile Learning," *World Conference on Mobile Learning (mLearn)*, Banff, Alberta, CANADA, October 2006.

56. J. Kurian, K. Sarac, and K. Almeroth, "Defending Network-Based Services Against Denial of Service Attacks," *International Conference on Computer Communication and Networks (IC3N)*, Arlington, Virginia, USA, October 2006.

55. A. Jardosh, K. Sanzgiri, E. Belding and K. Almeroth, "IQU: Practical Queue-Based User Association Management for WLANs--Case Studies, Architecture, and Implementation," ACM Mobicom, Marina del Rey, California, USA, September 2006.

54. C. Holman, K. Harras, and K. Almeroth, "A Proactive Data Bundling System for Intermittent Mobile Connections," IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON), Reston, Virginia, USA, September 2006.

53. G. Banks, M. Cova, V. Felmetsger, K. Almeroth, R. Kemmerer and G. Vigna, "SNOOZE: toward a Stateful NetwOrk prOtocol fuzZEr," *Information Security Conference (ISC)*, Samos Island, GREECE, September 2006.

52. K. Harras and K. Almeroth, "Inter-Regional Messenger Scheduling in Delay Tolerant Mobile Networks," *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Niagara Falls, New York, USA, June 2006.

51. M. Bulger, R. Mayer, and K. Almeroth, "Engaged By Design: Using Simulation to Promote Active Learning," **Outstanding Paper** at the *World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED MEDIA)*, Orlando, Florida, USA, June 2006.

50. A. Knight, K. Almeroth, R. Mayer, D. Chun, and B. Bimber, "Observations and Recommendations for Using Technology to Extend Interaction," *World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED MEDIA)*, Orlando, Florida, USA, June 2006.

49. H. Zhang, and K. Almeroth, "A Simple Classroom Network Access Control System," *World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED MEDIA)*, Orlando, Florida, USA, June 2006.

48. K. Harras and K. Almeroth, "Transport Layer Issues in Delay Tolerant Mobile Networks," *IFIP Networking Conference*, Coimbra, PORTUGAL, May 2006.

47. R. Mayer, A. Stull, J. Campbell, K. Almeroth, B. Bimber, D. Chun and A. Knight, "Some Shortcomings of Soliciting Students' Self-Reported SAT Scores," *American Educational Research Association (AERA) Annual Conference*, San Francisco, California, USA, April 2006.

46. K. Ramachandran, E. Belding, K. Almeroth, and M. Buddhikot, "Interference-Aware Channel Assignment in Multi-Radio Wireless Mesh Networks," *IEEE Infocom*, Barcelona, SPAIN, April 2006.

45. A. Jardosh, K. Ramachandran, K. Almeroth, and E. Belding, "Understanding Congestion in IEEE 802.11b Wireless Networks," *ACM/USENIX Internet Measurement Conference (IMC)*, Berkeley, California, USA, October 2005.

44. H. Zhang, K. Almeroth and M. Bulger, "An Activity Monitoring System to Support Classroom Research," *World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED MEDIA)*, Montreal, Quebec, CANADA, pp. 1444-1449, June 2005.

43. Z. Xiang, H. Zhang, J. Huang, S. Song and K. Almeroth, "A Hidden Environment Model for Constructing Indoor Radio Maps," *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Taormina, ITALY, June 2005.

42. K. Harras, K. Almeroth and E. Belding, "Delay Tolerant Mobile Networks (DTMNs): Controlled Flooding in Sparse Mobile Networks," *IFIP Networking Conference*, Waterloo, Ontario, CANADA, May 2005.

41. A. Garyfalos and K. Almeroth, "Coupons: Wide Scale Information Distribution for Wireless Ad Hoc Networks," *IEEE Global Telecommunications Conference (Globecom) Global Internet and Next Generation Networks Symposium*, Dallas, Texas, USA, pp. 1655-1659, December 2004.

40. A. Knight and K. Almeroth, "DeCAF: A Digital Classroom Application Framework," *IASTED International Conference on Communications, Internet and Information Technology (CIIT)*, St. Thomas, US Virgin Islands, November 2004.

39. P. Namburi, K. Sarac and K. Almeroth, "SSM-Ping: A Ping Utility for Source Specific Multicast," *IASTED International Conference on Communications, Internet and Information Technology (CIIT)*, St. Thomas, US Virgin Islands, November 2004.

38. K. Ramachandran, E. Belding and K. Almeroth, "DAMON: A Distributed Architecture for Monitoring Multi-hop Mobile Networks," *IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, Santa Clara, California, USA, October 2004.

37. A. Garyfalos and K. Almeroth, "Coupon Based Incentive Systems and the Implications of Equilibrium Theory," *IEEE Conference on Electronic Commerce (CEC)*, San Diego, California, USA, pp. 213-220, July 2004.

36. A. Knight, K. Almeroth and B. Bimber, "An Automated System for Plagiarism Detection Using the Internet," *World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED MEDIA)*, Lugano, Switzerland, pp. 3619-3625, June 2004.

35. H. Zhang and K. Almeroth, "Supplement to Distance Learning: Design for a Remote TA Support System," *World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED MEDIA)*, Lugano, Switzerland, pp. 2821-2830, June 2004.

34. U. Mohan, K. Almeroth and E. Belding, "Scalable Service Discovery in Mobile Ad hoc Networks," *IFIP Networking Conference*, Athens, Greece, pp. 137-149, May 2004.

33. V. Thanedar, K. Almeroth and E. Belding, "A Lightweight Content Replication Scheme for Mobile Ad hoc Environments," *IFIP Networking Conference*, Athens, Greece, pp. 125-136, May 2004.

32. R. Chalmers and K. Almeroth, "A Mobility Gateway for Small-Device Networks," *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, Orlando, Florida, USA, March 2004.

31. A. Jardosh, E. Belding, K. Almeroth and S. Suri, "Towards Realistic Mobility Models For Mobile Ad hoc Networks," *ACM Mobicom*, San Diego, California, USA, September 2003.

30. K. Sarac, P. Namburi and K. Almeroth, "SSM Extensions: Network Layer Support for Multiple Senders in SSM," *International Conference on Computer Communication and Networks (IC3N)*, Dallas, Texas, USA, October 2003.

29. K. Ramachandran and K. Almeroth, "MAFIA: A Multicast Management Solution for Access Control and Traffic Filtering," *IEEE/IFIP Conference on Management of Multimedia Networks and Services (MMNS)*, Belfast, Northern Ireland, September 2003.

28. J. Humfrey, S. Rollins, K. Almeroth, and B. Bimber, "Managing Complexity in a Networked Learning Environment," *World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED MEDIA)*, Honolulu, Hawaii, USA, pp. 60-63, June 2003.

27. K. Almeroth, S. Rollins, Z. Shen, and B. Bimber, "Creating a Demarcation Point Between Content Production and Encoding in a Digital Classroom," *World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED MEDIA)*, Honolulu, Hawaii, USA, pp. 2-5, June 2003.

26. M. Kolsch, K. Kvilekval, and K. Almeroth, "Improving Speaker Training with Interactive Lectures," *World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED MEDIA)*, Honolulu, Hawaii, USA, June 2003.

25. P. Rajvaidya and K. Almeroth, "Analysis of Routing Characteristics in the Multicast Infrastructure," *IEEE Infocom*, San Francisco, California, USA, April 2003.

24. S. Rollins and K. Almeroth, "Pixie: A Jukebox Architecture to Support Efficient Peer Content Exchange," *ACM Multimedia*, Juan Les Pins, FRANCE, December 2002.

23. S. Rollins, R. Chalmers, J. Blanquer, and K. Almeroth, "The Active Information System(AIS): A Model for Developing Scalable Web Services," *IASTED International Conference on Internet and Multimedia Systems and Applications (IMSA)*, Kauai, Hawaii, USA, August 2002.

22. S. Rollins and K. Almeroth, "Seminal: Additive Semantic Content for Multimedia Streams," *IASTED International Conference on Internet and Multimedia Systems and Applications (IMSA)*, Kauai, Hawaii, USA, August 2002.

21. B. Stiller, K. Almeroth, J. Altmann, L. McKnight, and M. Ott, "Content Pricing in the Internet," *SPIE ITCom Conference on Internet Performance and Control of Network Systems (IPCNS)*, Boston, Massachusetts, USA, July 2002.

20. S. Jagannathan, J. Nayek, K. Almeroth and M. Hofmann, "A Model for Discovering Customer Value for E-Content," *ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, Edmonton, Alberta, CANADA, July 2002.

19. S. Rollins and K. Almeroth, "Deploying and Infrastructure for Technologically Enhanced Learning," **Outstanding Paper** at the *World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED MEDIA)*, Denver, Colorado, USA, pp. 1651-1656, June 2002.

18. P. Rajvaidya and K. Almeroth, "Building the Case for Distributed Global Multicast Monitoring," *Multimedia Computing and Networking (MMCN)*, San Jose, California, USA, January 2002.

17. S. Jagannathan and K. Almeroth, "An Adaptive Pricing Scheme for Content Delivery Systems," *IEEE Global Internet*, San Antonio, Texas, USA, November 2001.

16. K. Sarac and K. Almeroth, "Providing Scalable Many-to-One Feedback in Multicast Reachability Monitoring Systems," *IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS)*, Chicago, Illinois, USA, October 2001.

15. S. Jagannathan and K. Almeroth, "The Dynamics of Price, Revenue and System Utilization," *IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS)*, Chicago, Illinois, USA, October 2001.

14. A. Kanwar, K. Almeroth, S. Bhattacharyya, and M. Davy, "Enabling End-User Network Monitoring via the Multicast Consolidated Proxy Monitor," *SPIE ITCom Conference on Scalability and Traffic Control in IP Networks (STCIPN)*, Denver, Colorado, USA, August 2001.

13. S. Jagannathan and K. Almeroth, "Using Tree Topology for Multicast Congestion Control," *International Conference on Parallel Processing (ICPP)*, Valencia, SPAIN, September 2001.

12. P. Rajvaidya and K. Almeroth, "A Router-Based Technique for Monitoring the Next-Generation of Internet Multicast Protocols," *International Conference on Parallel Processing (ICPP)*, Valencia, SPAIN, September 2001.

11. R. Chalmers and K. Almeroth, "Modeling the Branching Characteristics and Efficiency Gains of Global Multicast Trees," *IEEE Infocom*, Anchorage, Alaska, USA, April 2001.

10. R. Chalmers and K. Almeroth, "Developing a Multicast Metric," *Global Internet*, San Francisco, California, USA, December 2000.

9. K. Sarac and K. Almeroth, "Monitoring Reachability in the Global Multicast Infrastructure," *IEEE International Conference on Network Protocols (ICNP)*, Osaka, JAPAN, November 2000.

8. K. Almeroth, "A Long-Term Analysis of Growth and Usage Patterns in the Multicast Backbone (MBone)," *IEEE INFOCOM*, Tel Aviv, ISRAEL, March 2000.

7. K. Almeroth, K. Obraczka and D. De Lucia, "A Lightweight Protocol for Interconnecting Heterogeneous Devices in Dynamic Environments," *IEEE International Conference on Multimedia Computing and Systems (ICMCS)*, Florence, ITALY, June 1999.

6. K. Almeroth and M. Ammar, "The Interactive Multimedia Jukebox (IMJ): A New Paradigm for the

On-Demand Delivery of Audio/Video," **Best Paper** at *the Seventh International World Wide Web Conference (WWW)*, Brisbane, AUSTRALIA, April 1998.

5. K. Almeroth, M. Ammar and Z. Fei, "Scalable Delivery of Web Pages Using Cyclic Best-Effort (UDP) Multicast," *IEEE INFOCOM*, San Francisco, California, USA, June 1998.

4. K. Almeroth and M. Ammar, "Delivering Popular Web Pages Using Cyclic Unreliable Multicast (Extended Abstract)," *SPIE Conference on Voice, Video and Data Communications*, Dallas, Texas, USA, November 1997.

3. K. Almeroth, A. Dan, D. Sitaram and W. Tetzlaff, "Long Term Resource Allocation in Video Delivery Systems," *IEEE INFOCOM*,Kobe, JAPAN, April 1997.

2. K. Almeroth and M. Ammar, "On the Performance of a Multicast Delivery Video-On-Demand Service with Discontinuous VCR Actions," *International Conference on Communications (ICC)*, Seattle, Washington, USA, June 1995.

1. K. Almeroth and M. Ammar, "A Scalable, Interactive Video-On-Demand Service Using Multicast Communication," *International Conference on Computer Communication and Networks (IC3N)*, San Francisco, California, USA, September 1994.

C. Workshop Papers (refereed)

34. M. Tavakolifard, J. Gulla, K. Almeroth, F. Hopfgartner, B. Kille, T. Plumbaum, A. Lommatzsch, T. Brodt, A. Bucko, and T. Heintz, "Workshop and Challenge on News Recommender Systems," *ACM RecSys News Recommender Systems (NRS) Workshop and Challange*, Hong Kong, CHINA, October 2013.

33. M. Tavakolifard, K. Almeroth, and J. Gulla, "Does Social Contact Matter? Modelling the Hidden Web of Trust Underlying Twitter," *ACM International Workshop on Social Recommender Systems (SRS)*, Rio de Janeiro, BRAZIL, May 2013.

32. D. Johnson, E. Belding, K. Almeroth and G. van Stam, "Internet Usage and Performance Analysis of a Rural Wireless Network in Macha, Zambia," *ACM Networked Systems for Developing Regions (NSDR) Workshop*, San Francisco, California, USA, June 2010.

31. D. Havey, R. Chertov, and K. Almeroth, "Wired Wireless Broadcast Emulation," *International Workshop on Wireless Network Measurement (WiNMee)*, Seoul, Korea, June 2009.

30. R. Raghavendra, P. Acharya, E. Belding, and K. Almeroth, "MeshMon: A Multi-Tiered Framework for Wireless Mesh Network Monitoring," *ACM Mobihoc Wireless of the Students, by the Students, for the Students Workshop (S3)*, New Orleans, Louisiana, USA, May 2009.

29. G. Swamynathan, C. Wilson, B. Boe, B. Zhao, and K. Almeroth, "Do Social Networks Improve e-Commerce: A Study on Social Marketplaces," *ACM Sigcomm Workshop on Online Social Networks (WOSN)*, Seattle, Washington, USA, August 2008.

28. R. Raghavendra, E. Belding, and K. Almeroth, "Antler: A Multi-Tiered Approach to Automated

Wireless Network Management," *IEEE Workshop on Automated Network Management (ANM)*, Phoenix, Arizona, USA, April 2008.

27. S. Karpinski, E. Belding, and K. Almeroth, "Towards Realistic Models of Wireless Workload," *International Workshop on Wireless Network Measurement (WiNMee)*, Limassol, CYPRUS, April 2007.

26. K. Harras, M. Wittie, K. Almeroth, and E. Belding, "ParaNets: A Parallel Network Architecture for Challenged Networks," *IEEE Workshop on Mobile Computing Systems and Applications (HotMobile)*, Tucson, Arizona, USA, February 2007.

25. H. Caituiro-Monge, K. Almeroth, M. del Mar Alvarez-Rohena, "Friend Relay: A Resource Sharing Framework for Mobile Wireless Devices," *ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots (WMASH)*, Los Angeles, California, September 2006.

24. G. Swamynathan, Ben Y. Zhao and K. Almeroth, "Exploring the Feasibility of Proactive Reputations," *International Workshop on Peer-to-Peer Systems (IPTPS)*, Santa Barbara, California, USA, February 2006.

23. G. Swamynathan, Ben Y. Zhao and K. Almeroth, "Decoupling Service and Feedback Trust in a Peer-to-Peer Reputation System," *International Workshop on Applications and Economics of Peer-to-Peer Systems (AEPP)*, Nanjing, CHINA, November 2005.

22. K. Ramachandran, M. Buddhikot, G. Chandranmenon, S. Miller, E. Belding, and K. Almeroth, "On the Design and Implementation of Infrastructure Mesh Networks," *IEEE Workshop on Wireless Mesh Networks (WiMesh)*, Santa Clara, California, USA, September 2005.

21. A. Jardosh, K. Ramachandran, K. Almeroth and E. Belding, "Understanding Link-Layer Behavior in Highly Congested IEEE 802.11b Wireless Networks," Sigcomm Workshop on Experimental Approaches to Wireless Network Design and Analysis (EWIND), Philadelphia, Pennsylvania, USA, August 2005.

20. A. Sen Mazumder, K. Almeroth and K. Sarac, "Facilitating Robust Multicast Group Management," *Network and Operating System Support for Digital Audio and Video (NOSSDAV)* , Skamania, Washington, USA, June 2005.

19. Y. Sun, I. Sheriff, E. Belding and K. Almeroth, "An Experimental Study of Multimedia Traffic Performance in Mesh Networks," MobiSys International Workshop on Wireless Traffic Measurements and Modeling (WitMeMo), Seattle, Washington, USA, June 2005.

18. K. Ramachandran, K. Almeroth and E. Belding, "A Framework for the Management of Large-Scale Wireless Network Testbeds," International Workshop on Wireless Network Measurement (WiNMee), Trentino, ITALY, April 2005.

17. A. Garyfalos, K. Almeroth and K. Sanzgiri, "Deployment Complexity Versus Performance Efficiency in Mobile Multicast," *International Workshop on Broadband Wireless Multimedia: Algorithms, Architectures and Applications (BroadWiM)*, San Jose, California, USA, October 2004.

16. C. Ho, K. Ramachandran, K. Almeroth and E. Belding, "A Scalable Framework for Wireless Network Monitoring," *ACM International Workshop on Wireless Mobile Applications and Services on WLAN*

Hotspots (WMASH), Philadelphia, Pennsylvania, USA, October 2004.

15. A. Garyfalos, K. Almeroth and J. Finney, "A Hybrid of Network and Application Layer Multicast for Mobile IPv6 Networks," *International Workshop on Large-Scale Group Communication (LSGC)*, Florence, ITALY, October 2003.

14. A. Garyfalos, K. Almeroth and J. Finney, "A Comparison of Network and Application Layer Multicast for Mobile IPv6 Networks," *ACM Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, San Diego, California, USA, September 2003.

13. S. Jagannathan, and K. Almeroth, "Pricing and Resource Provisioning for Delivering E-Content On-Demand with Multiple Levels-of-Service," *International Workshop on Internet Charging and QoS Technologies (ICQT)*, Zurich, SWITZERLAND, October 2002.

12. S. Rollins, K. Almeroth, D. Milojicic, and K. Nagaraja, "Power-Aware Data Management for Small Devices," *Workshop on Wireless Mobile Multimedia (WoWMoM)*, Atlanta, GA, USA, September 2002.

11. K. Almeroth, S. Bhattacharyya, and C. Diot, "Challenges of Integrating ASM and SSM IP Multicast Protocol Architectures," *International Workshop on Digital Communications: Evolutionary Trends of the Internet (IWDC)*, Taormina, ITALY, September 2001.

10. K. Sarac and K. Almeroth, "Scalable Techniques for Discovering Multicast Tree Topology," *Network and Operating System Support for Digital Audio and Video (NOSSDAV)* , Port Jefferson, New York, USA, June 2001.

9. P. Rajvaidya, K. Almeroth and K. Claffy, "A Scalable Architecture for Monitoring and Visualizing Multicast Statistics," *IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM)*, Austin, Texas, USA, December 2000.

8. S. Jagannathan, K. Almeroth and A. Acharya, "Topology Sensitive Congestion Control for Real-Time Multicast," *Network and Operating System Support for Digital Audio and Video (NOSSDAV)* , Chapel Hill, North Carolina, USA, June 2000.

7. K. Sarac and K. Almeroth, "Supporting the Need for Inter-Domain Multicast Reachability," *Network and Operating System Support for Digital Audio and Video (NOSSDAV)* , Chapel Hill , North Carolina, USA, June 2000.

6. D. Makofske and K. Almeroth, "MHealth: A Real-Time Multicast Tree Visualization and Monitoring Tool," *Network and Operating System Support for Digital Audio and Video (NOSSDAV)* , Basking Ridge New Jersey, USA, June 1999.

5. K. Almeroth and Y. Zhang, "Using Satellite Links as Delivery Paths in the Multicast Backbone (MBone)," *ACM/IEEE International Workshop on Satellite-Based Information Services (WOSBIS)*, Dallas, Texas, USA, October 1998.

4. M. Ammar, K. Almeroth, R. Clark and Z. Fei, "Multicast Delivery of Web Pages OR How to Make Web Servers Pushy," *Workshop on Internet Server Performance (WISP)*, Madison, Wisconsin, USA, June 1998.

3. K. Almeroth and M. Ammar, "Prototyping the Interactive Multimedia Jukebox," *Mini-conference on*

Multimedia Appliances, Interfaces, and Trials--International Conference on Communications (ICC), Montreal, Quebec, CANADA, June 1997.

2. K. Almeroth and M. Ammar, "Collection and Modeling of the Join/Leave Behavior of Multicast Group Members in the MBone," *High Performance Distributed Computing Focus Workshop (HPDC)*, Syracuse, New York, USA, August 1996.

1. K. Almeroth and M. Ammar, "The Role of Multicast Communication in the Provision of Scalable and Interactive Video-On-Demand Service," *Network and Operating System Support for Digital Audio and Video (NOSSDAV)* , Durham, New Hampshire, USA, April 1995.

D. Non-Refereed Publications

8. K. Almeroth, E. Belding, M. Buddhikot, G. Chandranmenon, S. Miller, and K. Ramachandran, "Infrastructure Mesh Networks," *U.S. Patent Application US20070070959 A1*, September 2005.

7. K. Almeroth, R. Caceres, A. Clark, R. Cole, N. Duffield, T. Friedman, K. Hedayat, K. Sarac, M. Westerlund, "RTP Control Protocol Extended Reports (RTCP XR)," *Internet Engineering Task Force (IETF) Request for Comments (RFC) 3611*, November 2003.

6. Z. Albanna, K. Almeroth, D. Meyer, and M. Schipper, "IANA Guidelines for IPv4 Multicast Address Allocation," *Internet Engineering Task Force (IETF) Request for Comments (RFC) 3171*, August 2001.

5. B. Quinn and K. Almeroth, "IP Multicast Applications: Challenges and Solutions," *Internet Engineering Task Force (IETF), Request for Comments (RFC) 3170*, September 2001.

4. K. Almeroth, L. Wei and D. Farinacci, "Multicast Reachability Monitor (MRM) Protocol," *Internet Engineering Task Force Internet Draft*, July 2000.

3. K. Almeroth and L. Wei, "Justification for and use of the Multicast Reachability Monitor (MRM) Protocol," *Internet Engineering Task Force Internet Draft*, March 1999.

2. K. Almeroth, "Managing IP Multicast Traffic: A First Look at the Issues, Tools, and Challenges," IP Multicast Initiative White Paper, San Jose, California, USA, February 1999.

1. K. Almeroth, K. Obraczka and D. De Lucia, "Pseudo-IP: Providing a Thin Network Protocol for Semi-Intelligent Wireless Devices," *DARPA/NIST Smart Spaces Workshop*, Gaithersburg, Maryland, USA, July 1998.

E. Released Software Systems

19. ***A Multi-radio Wireless Mesh Network Architecture -- http://moment.cs.ucsb.edu/tic/.*** Released December 1, 2006 (with K. Ramachandran, I. Sheriff, and E. Belding). The software as part of a multi-radio wireless mesh network that includes a Split Wireless Router that alleviates the interference that can occur between commodity radios within a single piece of hardware. The second is server software to perform channel assignment and communicate the assignments throughout the mesh

network.

18. ***AODV-Spanning Tree (AODV-ST) -- http://www.cs.ucsb.edu/~krishna/aodv-st/.*** Released September 1, 2006 (with K. Ramachandran and E. Belding). AODV-ST is an extension of the well-known AODV protocol specifically designed for wireless mesh networks. The advantages of AODV-ST over AODV include support for high throughput routing metrics, automatic route maintenance for common-case traffic, and low route discovery latency.

17. ***The Multicast Detective -- http://www.nmsl.cs.ucsb.edu/mcast_detective/.*** Released September 1, 2005 (with A. Sen Mazumder). The multicast detective is a robust solution to determine the existence and nature of multicast service for a particular user. By performing a series of tests, a user can determine whether there is network support for multicast, and consequently, whether a multicast group join is likely to succeed.

16. ***AutoCap: Automatic and Accurate Captioning -- http://www.nmsl.cs.ucsb.edu/autocap/.*** Released August 1, 2005 (with A. Knight). AutoCap is a software system that takes as input an audio/video file and a text transcript. AutoCap creates captions by aligning the utterances in the audio/video file to the transcript. For those words that are not recognized, AutoCap estimates when the words were spoken along with an error bound that gives the content creator an idea of caption accuracy. The result is a collection of accurately time-stamped captions that can be displayed with the video.

15. ***PAIRwise Plagiarism Detection System -- http://cits.ucsb.edu/pair/.*** Released July 1, 2005 (with A. Knight). PAIRwise is a plagiarism detection system with: (1) an easy-to-use interface for submitting papers, (2) a flexible comparison engine that allows intra-class, inter-class, and Internet-based comparisons, and (3) an intuitive graphical presentation of results.

14. ***DAMON Multi-Hop Wireless Network Monitoring -- http://moment.cs.ucsb.edu/damon/.*** Released October 1, 2004 (with K. Ramachandran and E. Belding). DAMON is a distributed system for monitoring multi-hop mobile networks. DAMON uses agents within the network to monitor network behavior and send collected measurements to data repositories. DAMON's generic architecture supports the monitoring of a wide range of protocol, device, or network parameters.

13. ***Multicast Firewall -- http://www.nmsl.cs.ucsb.edu/mafia/.*** Released June 1, 2004 (with K. Ramachandran). MAFIA, a multicast firewall and traffic management solution, has the specific aim of strengthening multicast security through multicast access control, multicast traffic filtering, and DoS attack prevention.

12. ***AODV@IETF Peer Routing Software-- http://moment.cs.ucsb.edu/aodv-ietf/.*** Released November 1, 2003 (with K. Ramachandran and E. Belding). One of the first large-scale efforts to run the Ad hoc On demand Distance Vector (AODV) routing protocol in a public space (at the Internet Engineering Task Force (IETF)). The implementation includes a daemon that runs on both the Linux and Windows operating systems.

11. ***Mobility Obstacles -- http://moment.cs.ucsb.edu/mobility/.*** Released September 1, 2003 (with A. Jardosh, E. Belding, and S. Suri). The topology and movement of nodes in ad hoc protocol simulation are key factors in protocol performance. In this project, we have developed ns-2 simulation plug-ins that create more realistic movement models through the incorporation of obstacles. These obstacles are utilized to restrict both node movement and wireless transmissions.

10. ***mwalk -- http://www.nmsl.cs.ucsb.edu/mwalk/.*** Released December 1, 2000 (with R. Chalmers). Mwalk is a collection of Java applications and Perl scripts which re-create a global view of a multicast session from mtrace and RTCP logs. Users to the site can download mwalk, examine the results of our analysis, or download data sets for use in simulations dependent on multicast tree characteristics.

9. ***MANTRA2 -- http://www.nmsl.cs.ucsb.edu/mantra/.*** Released December 1, 1999 (with P. Rajvaidya). This new version of MANTRA focuses on the visualization of inter-domain routing statistics. Working in conjunction with the Cooperative Association for Internet Data Analysis (CAIDA) we have developed advanced collection and visualization techniques.

8. ***MRM -- http://www.nmsl.cs.ucsb.edu/mrm/.*** Released October 1, 1999 (with K. Sarac). MRM is the Multicast Reachability Protocol. We have implemented an end-host agent that responds to MRM Manager commands. Our end-host agent works in conjunction with Cisco routers to detect and isolate multicast faults.

7. ***MANTRA -- http://www.nmsl.cs.ucsb.edu/mantra/.*** Released January 1, 1999 (with P. Rajvaidya). MANTRA is the Monitoring and Analysis of Traffic in Multicast Routers. It uses scripts to collect and display data from backbone multicast routers.

6. ***SDR Monitor -- http://www.nmsl.cs.ucsb.edu/sdr-monitor/.*** Released January 1, 1999 (with K. Sarac). The SDR Monitor receives e-mail updates from participants containing information about observed sessions in the MBone. A global view of multicast reachability is then constructed.

5. ***The MHealth tool -- http://www.nmsl.cs.ucsb.edu/mhealth/.*** Released September 1, 1998 (with D. Makofske). The mhealth tool graphically visualizes MBone multicast group trees and provides `health' information including end-to-end losses per receiver and losses on a per hop basis. The implementation required expertise in Java, the MBone tools, and Unix.

4. ***The MControl tool -- http://www.nmsl.cs.ucsb.edu/mcontrol/.*** Released August 1, 1998 (with D. Makofske). Mcontrol is a tool to provide VCR-based interactivity for live MBone sessions. The implementation required expertise in Java, the MBone tools, and Unix.

3. ***Interactive Multimedia Jukebox (IMJ) -- http://imj.ucsb.edu/.*** Released October 1, 1996. The IMJ combines the WWW and the MBone conferencing tools to provide a multi-channel video jukebox offering both instructional and entertainment programming on a wide scale. The implementation required expertise in HTML, Perl, C, the MBone tools, and Unix.

2. ***Mlisten -- http://www.cc.gatech.edu/computing/Telecomm/mbone/.*** Released September 1, 1995. A tool to continuously collect MBone multicast group membership information including number and location of members, membership duration, and inter-arrival time for all audio and video sessions. The implementation required expertise in C, Tcl/Tk, the MBone tools, and UNIX socket programming.

1. ***Audio-on-Demand (AoD)***. March 1, 1995. A server/client prototype to demonstrate interactivity in near VoD systems. The AoD server provides songs-on-demand and VCR-like functions via multicast IP over Ethernet. The implementation required expertise in C, OpenWindows programming, UNIX socket programming, and network programming.

F. Tutorials, Panels and Invited Talks

- "Sensing and Opportunistic Delivery of Ubiquitous Video in Health Monitoring, On-Campus and Social Network Applications," Workshop on Mobile Video Delivery (MoViD), Chapel Hill North Carolina, USA, February 2012.
- "Medium Access in new Contexts: Reinventing the Wheel?," USC Invited Workshop on Theory and Practice in Wireless Networks, Los Angeles, California, USA, May 2008.
- "The Wild, Wild West: Wireless Networks Need a New Sheriff," University of Florida CISE Department Lecture Series, Gainesville, Florida, USA, February 2008.
- "Distinguishing Between Connectivity, Intermittent Connectivity, and Intermittent Disconnectivity," Keynote at the ACM MobiCom Workshop on Challenged Networks (CHANTS), Montreal, CANADA, September 2007.
- "The Three Ghosts of Multicast: Past, Present, and Future," Keynote at the Trans-European Research and Education Networking Association (TERENA) Networking Conference, Lynby, DENMARK, May 2007.
- "Multicast Help Wanted: From Where and How Much?," Keynote at the Workshop on Peer-to-Peer Multicasting (P2PM), Las Vegas, Nevada, USA, January 2007.
- "The Confluence of Wi-Fi and Apps: What to Expect Next," Engineering Insights, UC-Santa Barbara, Santa Barbara, California, USA, October 2006.
- "Challenges, Opportunities, and Implications for the Future Internet," University of Minnesota Digital Technology Center, Minneapolis, Minnesota, USA, September 2006.
- "Wireless Technology as a Catalyst: Possibilities for Next-Generation Interaction," Santa Barbara Forum on Digital Transitions, Santa Barbara, California, USA, April 2006.
- "Challenges and Opportunities in an Internet with Pervasive Wireless Access," University of Texas--Dallas Computer Science Colloquium, Dallas, Texas, USA, March 2006.
- "Challenges and Opportunities with Pervasive Wireless in the Internet," Duke University Computer Science Colloquium, Durham, North Carolina, USA, February 2006.
- "The Span From Wireless Protocols to Social Applications," Intel Research Labs, Cambridge, United Kingdom, December 2005.
- "The Internet Dot.Com Bomb and Beyond the Dot.Com Calm," CSE IGERT and Cal Poly Lecture Series, San Luis Obispo, California, USA, October 2005.
- "Panel: Directions in Networking Research," IEEE Computer Communications Workshop (CCW), Irvine, California, USA, October 2005.
- "Economic Incentives for Ad Hoc Networks," KAIST New Applications Seminar, Seoul, South Korea, March 2004.

- "New Applications for the Next Generation Internet," Citrix Systems, Santa Barbara, California, USA, March 2004.
- "PI: The Imperfect Pursuit of Pure Pattern," CITS Visions in Technology Series, Santa Barbara, California, USA, January 2004.
- "Panel: Core Networking Issues and Protocols for the Internet," National Science Foundation (NSF) Division of Advanced Networking Infrastructure and Research (ANIR) Principal Investigators Workshop, Washington DC, USA, March 2003.
- "Panel: Pricing for Content in the Internet," SPIE ITCom Internet Performance and Control of Network Systems, Boston, Massachusetts, USA, July 2002.
- "The Technology Behind Wireless LANs," Central Coast MIT Enterprise Forum, Santa Barbara, California, USA, March 2002.
- "Lessons Learned in the Digital Classroom," Center for Information and Technology Brown Bag Symposium, Santa Barbara, California, USA, March 2002.
- "The Evolution of Advanced Networking Services: From the ARPAnet to Internet2," California State University--San Luis Obispo CS Centennial Colloquium Series, San Luis Obispo, California, USA, February 2002.
- "Deployment of IP Multicast in Campus Infrastructures," Internet2 Campus Deployment Workshop, Atlanta, Georgia, USA, May 2001.
- "Multicast: Is There Anything Else to Do?," Sprint Research Retreat, Miami, Florida, USA, May 2001.
- "The Evolution of Next-Generation Internet Services and Applications," Government Technology Conference 2001 (GTC) for the Western Region, Sacramento, California, USA, May 2001.
- "I2 Multicast: Not WIDE-scale Deployment, FULL-scale Deployment," Closing Plenary, Internet2 Member Meetings, Washington, D.C., USA, March 2001.
- "Panel: Beyond IP Multicast," Content Delivery Networks (CDN), New York, New York, USA, February 2001.
- "Viable Multicast Pricing & Business Models for Wider-Scale Deployment," Content Delivery Networks (CDN), New York, New York, USA, February 2001.
- "IP Multicast: Modern Protocols, Deployment, and Management," Content Delivery Networks (CDN), New York, New York, USA, February 2001 & San Jose, California, USA, December 2001.
- "Under the Hood of the Internet," Technology 101: Technology for Investors, Center for Entrepreneurship & Engineering Management, November 2000.
- "Understanding Multicast Traffic in the Internet," (1) University of Virginia, (2) University of Maryland, and (3) Columbia University, September 2000.
- "The Bad, The Ugly, and The Good: The Past, Present, and Future of Multicast," Digital Fountain, San Francisco, California, USA, August 2000.

- "Implications of Source-Specific Multicast (SSM) on the Future of Internet Content Delivery," Occam Networks, Santa Barbara, California, USA, August 2000.
- "Introduction to Multicast Routing Protocols," UC-Berkeley Open Mash Multicast Workshop, Berkeley, California, USA, July 2000.
- "Efforts to Understand Traffic and Tree Characteristics," University of Massachusetts--Amherst Colloquia, Amherst, Massachusetts, USA, May 2000.
- "Monitoring Multicast Traffic," Sprint Research Retreat, Half Moon Bay, California, USA, April 2000.
- "What is the Next Generation of Multicast in the Internet?," HRL Laboratories, Malibu, California, USA, January 2000.
- "Mission and Status of the Center for Information Technology and Society (CITS)," Intel Research Council, Portland, Oregon, USA, September 1999.
- "Multicast at a Crossroads," IP Multicast Initiative Summits and Bandwidth Management Workshops, San Francisco, CA, USA, (1) October 1999; (2) February 2000; and (3) June 2000.
- "IP Multicast: Modern Protocols, Deployment, and Management," Networld+Interop: (1) Las Vegas, Nevada, USA--May 2000; (2) Tokyo, JAPAN--June 2000; (3) Atlanta, Georgia, USA--September 2000; (4) Las Vegas, Nevada, USA--May 2001; (5) Las Vegas, Nevada, USA--May 2002.
- "IP Multicast: Practice and Theory" (w/ Steve Deering), Networld+Interop: (1) Las Vegas, Nevada, USA--May 1999; (2) Tokyo, JAPAN--June 1999; and (3) Atlanta, Georgia, USA--September 1999.
- "Internet2 Multicast Testbeds and Applications," Workshop on Protocols for High Speed Networks (PfHSN), Salem, Massachusetts, USA, August 1999.
- "IP Multicast: Protocols for the Intra- and Inter-Domain," Lucent Technologies, Westford, Massachusetts, USA, August 1999.
- "Internet2 Multicast Testbeds and Applications," NASA Workshop: Bridging the Gap, Moffett Field, California, USA, August 1999.
- "The Evolution of Next-Generation Services and Applications in the Internet," Tektronix Distinguished Lecture Series, Portland, Oregon, USA, May 1999.
- "Multicast Applications and Infrastructure in the Next Generation Internet," CENIC 99 Workshop on Achieving Critical Mass for Advanced Applications, Monterey, California, USA, May 1999.
- "Multicast Traffic Monitoring and Analysis Work at UCSB" (w/ P. Rajvaidya), Workshop on Internet Statistics and Metrics Analysis (ISMA), San Diego, California, USA, April 1999.
- "How the Internet Works: Following Bits Around the World," Science Lite, Santa Barbara General Affiliates and Office of Community Relations, California, USA, February 1999.
- "Managing Multicast: Challenges, Tools, and the Future," IP Multicast Initiative Summit, San Jose, California, USA, February 1999.

- "The Future of Multicast Communication and Protocols," Internet Bandwidth Management Summit (iBAND), San Jose, California, USA, November 1998.
- "An Overview of IP Multicast: Applications and Deployment," (1) Workshop on Evaluating IP Multicast as the Solution for Webcasting Real-Time Networked Multimedia Applications, New York, New York, USA, July 1998; and (2) Satellites and the Internet Conference, Washington, D.C., USA, July 1998.
- "IETF Developments in IP Multicast," IP Multicast Initiative Summit, San Jose, California, USA, February 1998.
- "An Introduction to IP Multicast and the Multicast Backbone (MBone)" vBNS Technical Meeting sponsored by the National Center for Network Engineering (NLANR), San Diego, California, USA, February 1998.
- "Using Multicast Communication to Deliver WWW Pages" Computer Communications Workshop (CCW '97), Phoenix, Arizona, USA, September 1997.

G. Research Funding

- K. Almeroth, "Packet Scheduling Using IP Embedded Transport Instrumentation," Cisco Systems Inc., $100,000, 3/1/13-8/31/14.
- K. Almeroth, E. Belding and S.J. Lee, "GOALI: Maximizing Available Bandwidth in Next Generation WLANs", National Science Foundation (NSF), $101,088, 10/1/13-9/30/14.
- K. Almeroth and E. Belding, "GOALI: Intelligent Channel Management in 802.11n Networks," National Science Foundation (NSF), $51,000, 10/1/10-9/30/11.
- B. Zhao, K. Almeroth, H. Zheng, and E. Belding, "NeTS: Medium: Airlab: Distributed Infrastructure for Wireless Measurements," National Science Foundation (NSF), $700,000, 9/1/09-8/13/13.
- K. Almeroth, E. Belding and T. Hollerer, "NeTS-WN: Wireless Network Health: Real-Time Diagnosis, Adaptation, and Management," National Science Foundation (NSF), $600,000, 10/1/07-9/30/10.
- K. Almeroth, "Next-Generation Service Engineering in Internet2," University Consortium for Advanced Internet Development (UCAID), $1,254,000, 7/1/04-6/30/09 (reviewed and renewed each year).
- B. Manjunath, K. Almeroth, F. Bullo, J. Hespanha, T. Hollerer, C. Krintz, U. Madhow, K. Rose, A. Singh, and M. Turk, "Large-Scale Multimodal Wireless Sensor Network," Office of Naval Research Defense University Research Instrumentation Program (DURIP), $655,174, 4/14/08-4/14/09.
- K. Almeroth and E. Belding, "Improving Robustness in Evolving Wireless Infrastructures," Intel Corporation, $135,000, 7/1/06-6/30/09 (reviewed and renewed for second and third year).
- K. Almeroth and K. Sarac, "Bridging Support in Mixed Deployment Multicast Environments," Cisco Systems Inc., $100,000, 9/1/07-8/31/08.

- K. Sarac and K. Almeroth, "Building the Final Piece in One-to-Many Content Distribution," Cisco Systems Inc., $95,000, 9/1/06-8/31/07.

- E. Belding, K. Almeroth and J. Gibson, "Real-Time Communication Support in a Ubiquitous Next-Generation Internet," National Science Foundation (NSF), $900,000, 10/1/04-9/30/07.

- K. Almeroth and K. Sarac, "Improving the Robustness of Multicast in the Internet," Cisco Systems Inc., $80,000, 9/1/04-8/31/05.

- R. Mayer, B. Bimber, K. Almeroth and D. Chun, "Assessing the Pedagogical Implications of Technology in College Courses," Mellon Foundation, $350,000, 7/1/04-6/30/07.

- B. Bimber, A. Flanagin and C. Stol, "Technological Change and Collective Association: Changing Relationships Among Technology, Organizations, Society and the Citizenry," National Science Foundation (NSF), $329,175, 7/1/04-6/30/07.

- K. Almeroth and B. Bimber, "Plagiarism Detection Techniques and Software," UCSB Instructional Development, $22,000, 7/1/04-6/30/05.

- K. Almeroth, "Student Travel Support for the 14th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)," National Science Foundation (NSF), $10,000, 5/1/04-8/31/04.

- K. Almeroth, "An Automated Indexing System for Remote, Archived Presentations," QAD Inc., $25,000, 5/1/04-6/30/05.

- K. Almeroth and M. Turk, "A Remote Teaching Assistant Support System," Microsoft, $40,000, 1/1/04-6/30/05.

- K. Almeroth, "Supporting Multicast Service Functionality in Helix," Real Networks, $30,000, 9/1/03-6/30/04.

- K. Almeroth and E. Belding, "Service Discovery in Mobile Networks," Nokia Summer Research Grant (U. Mohan), $10,240, 7/1/03-9/30/03.

- K. Almeroth, D. Zappala, "Building a Global Multicast Service," Cisco Systems Inc., $100,000, 1/1/03-indefinite.

- K. Almeroth, "Developing A Dynamic Protocol for Candidate Access Router Discovery," Nokia Graduate Student Fellowship (R. Chalmers), $26,110, 9/01/02-6/30/03.

- B. Bimber and K. Almeroth, "The Role of Collaborative Groupware in Organizations," Toole Family Foundation, $182,500 ($20,000 cash plus $162,500 in software), 9/1/02-indefinite.

- B. Manjunath, et al., "Digital Multimedia: Graduate Training Program in Interactive Digital Multimedia," National Science Foundation (NSF), $2,629,373, 4/1/02-3/31/07.

- J. Green, K. Almeroth, et al., "Inquiry in the Online Context: Learning from the Past, Informing the Future," UCSB Research Across Disciplines, $10,000, 9/1/01-8/31/02.

- K. Almeroth, "Monitoring and Maintaining the Global Multicast Infrastructure," Cisco Systems Inc.,

$54,600, 7/1/01-indefinite.

- R. Kemmerer, K. Almeroth, et al., "Hi-DRA High-speed, Wide-area Network Detection, Response, and Analysis," Department of Defense (DoD), $4,283,500, 5/1/01-4/30/06.
- A. Singh, K. Almeroth, et al., "Digital Campus: Scalable Information Services on a Campus-wide Wireless Network," National Science Foundation (NSF), 1,450,000, 9/15/00-12/31/04.
- K. Almeroth, "Visualizing the Global Multicast Infrastructure," UC MICRO w/ Cisco Systems Inc., $85,438, 7/1/00-6/30/02.
- H. Lee, K. Almeroth, et al., "Dynamic Sensing Systems," International Telemetering Foundation, $260,000, 07/01/00-06/30/04.
- B. Bimber and K. Almeroth, "Funding for the Center on Information Technology and Society," $250,000 from Dialogic (an Intel Company) and $250,000 from Canadian Pacific.
- K. Almeroth, "CAREER: From Protocol Support to Applications: Elevating Multicast to a Ubiquitous Network Service," National Science Foundation (NSF), $200,000, 9/1/00-8/31/04.
- K. Almeroth, "Characterizing Multicast Use and Efficiency in the Inter-Domain," Sprint Advanced Technology Laboratories, $62,500, 3/1/00-indefinite.
- K. Almeroth, "Producing the Next Generation of Multicast Monitoring and Management Protocols and Tools," UC MICRO w/ Cisco Systems Inc., $124,500, 7/1/99 - 6/30/01.
- K. Almeroth, "Utilizing Satellite Links in the Provision of an Inter-Wide Multicast Service," HRL Laboratories, $20,000, 7/1/99 - indefinite.
- T. Smith, K. Almeroth, et al., "Alexandria Digital Earth Prototype," National Science Foundation, $5,400,000, 4/1/99-3/31/04.
- V. Vesna, K. Almeroth, et al., "Online Public Spaces: Multidisciplinary Explorations in Multi-User Environments (OPS:MEME), Phase II," UCSB Research Across Disciplines, $50,000, 9/1/98-8/31/99.
- K. Almeroth, "Techniques and Analysis for the Provision of Multicast Route Management," UC MICRO w/ Cisco Systems Inc., $97,610, 7/1/98 - 6/30/00.
- K. Almeroth, "Capturing and Modeling Multicast Group Membership in the Multicast Backbone (MBone)," UC MICRO w/ Hughes Research Labs, $19,146, 7/1/98 - 12/31/99.
- K. Almeroth, "Building a Content Server for the Next Generation Digital Classroom," UCSB Faculty Research Grant, $5,000, 7/1/98-6/31/99.

H. Research Honors and Awards

- IEEE Fellow Status, 2013
- Finalist for Best Paper Award, IEEE Conference on Sensor and Ad Hoc Communications and

Networks (SECON), June 2008
- Best Paper Award, Passive and Active Measurement (PAM) Conference, April 2007
- Outstanding Paper Award, World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED MEDIA), June 2006
- IEEE Senior Member Status, 2003
- Finalist for Best Student Paper Award, ACM Multimedia, December 2002
- Outstanding Paper Award, World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED MEDIA), June 2002
- Computing Research Association (CRA) Digital Government Fellowship, 2001
- National Science Foundation CAREER Award, 2000
- Best Paper Award, 7th International World Wide Web Conference, April 1998

III. Service

A. Professional Activities

1. Society Memberships

Member, Association for Computing Machinery (ACM): 1993-present
Member, ACM Special Interest Group on Communications (SIGComm): 1993-present
Fellow, Institute of Electrical and Electronics Engineers (IEEE): 1993-present
Member, IEEE Communications Society (IEEE ComSoc): 1993-present
Member, American Society for Engineering Education (ASEE): 2003-2006

2. Review Work for Technical Journals and Publishers

NSF CISE research proposals, IEEE/ACM Transactions on Networking, IEEE/ACM Transactions on Computers, IEEE/ACM Transactions on Communications, IEEE Transactions on Circuits and Systems for Video Technology, IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Multimedia, IEEE Communications, IEEE Communications Letters, IEEE Network, IEEE Internet Computing, IEEE Multimedia, IEEE Aerospace & Electronics Systems Magazine, ACM Transactions on Internet Technology, ACM Transactions on Multimedia Computing, Communications and Applications, ACM Computing Surveys, ACM Computer Communications Review, ACM Computeres in Entertainment, ACM/Springer Multimedia Systems Journal, AACE Journal of Interactive Learning (JILR), International Journal of Computer Mathematics, Journal of Communications and Networks, Journal of Parallel and Distributed Computing, Journal of Network and Systems Management, Journal of High Speed Networking, Journal of Communications and Networks, Journal on Selected Areas in Communications, Journal of Wireless Personal Communications, Personal Mobile Communications, Annals of Telecommunications, International Journal of Wireless and Mobile Computing, Pervasive and Mobile Computing (PMC), Wireless Networks Journal, Computer

Networks Journal, Cluster Computing, Computer Communications, Mobile Computing and Communications Review, Performance Evaluation, Software--Practice & Experience, Information Processing Letters, ACM Sigcomm, ACM Multimedia, ACM Network and System Support for Digital Audio and Video Workshop (NOSSDAV), ACM Sigcomm Workshop on the Economics of Peer-to-Peer Systems (P2PEcon), ACM Sigcomm Workshop on Challenged Networks (CHANTS), IEEE Infocom, IEEE Globecom, IEEE Global Internet (GI) Symposium, IEEE Globecom Automatic Internet Symposium, IEEE Globecom Internet Services and Enabling Technologies (IS&ET) Symposium, IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), IEEE International Conference on Network Protocols (ICNP), IEEE Conference on Sensor and Ad Hoc Communications and Networks (SECON), IEEE International Conference on Multimedia and Exposition (ICME), IEEE International Conference on Communications (ICC), IEEE International Conference on Parallel and Distributed Systems (ICPADS) IEEE International Symposium on High-Performance Distributed Computing (HPDC), IEEE International Conference on Distributed Computing Systems (ICDCS), IEEE International Workshop on Quality of Service (IWQoS), IEEE/IFIP Network Operations and Management Symposium (NOMS), IFIP/IEEE International Symposium on Integrated Network Management (IM), IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS), IEEE Aerospace & Electronics Systems Magazine, SPIE Conference on Multimedia Computing and Networking (MMCN), IFIP Networking, IASTED International Conference on Information Systems and Databases (ISD), IASTED International Conference on Communications, Internet, and Information Technology, IASTED International Conference on Internet and Multimedia Systems and Applications (IMSA), IASTED International Conference on European Internet and Multimedia Systems and Applications (EuroIMSA), IASTED International Conference on Communications and Computer Networks (CCN), IASTED International Conference on Software Engineering and Applications (SEA), International Conference on Computer and Information Science (ICIS), International Association for Development of the Information Society (IADIS) International Conference on the WWW/Internet, Workshop on Network Group Communication (NGC), International Conference on Next Generation Communication (CoNEXT), International Conference on Parallel Processing (ICPP), International Conference on Computer Communications and Networks (IC3N), International Workshop on Hot Topics in Peer-to-Peer Systems (Hot-P2P), International Workshop on Wireless Network Measurements (WiNMee), International Workshop on Incentive-Based Computing (IBC), International Workshop on Multi-hop Ad Hoc Networks (REALMAN), International Workshop on Broadband Wireless Multimedia: Algorithms, Architectures and Applications (BroadWIM), International Packet Video (PV) Workshop, High Performance Networking Conference (HPN), International Parallel Processing Symposium (IPPS), International Symposium on Innovation in Information & Communication Technology (ISIICT), Workshop on Coordinated Quality of Service in Distributed Systems (COQODS), Pearson Education (Cisco Press) Publishers, Macmillan Technical Publishing, and Prentice Hall Publishers.

3. Conference Committee Activities

Journal/Magazine Editorial Board

IEEE/ACM Transactions on Networking (ToN): 2003-2009, 2013-present
Journal of Network and Systems Management (JNSM): 2011-present

ACM Computers in Entertainment: 2002-present
IEEE Network: 1999-2012
AACE Journal of Interactive Learning Research (JILR): 2003-2012
IEEE Transactions on Mobile Computing (TMC): 2006-2011
ACM Computer Communications Review (CCR): 2006-2010

Journal/Magazine Guest Editorship

IEEE Journal on Selected Areas in Communications (JSAC) Special Issue on "Delay and Disruption Tolerant Wireless Communication", June 2008
Computer Communications Special Issue on "Monitoring and Measuring IP Networks", Summer 2005
Computer Communications Special Issue on "Integrating Multicast into the Internet", March 2001

Conference/Workshop Steering Committee

IEEE International Conference on Network Protocols (ICNP): 2007-present
ACM Sigcomm Workshop on Challenged Networks (CHANTS): 2006-present
International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV): 2001-present, 2005-2011 (chair), 2012-present (co-chair)
IEEE Global Internet (GI) Symposium: 2005-2013
IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS): 2005-2009

Conference/Workshop Chair

International Conference on Communication Systems and Networks (COMSNETS): 2014 (co-chair)
ACM International Conference on Next Generation Communication (CoNext): 2013 (co-chair)
ACM RecSys News Recommender Systems (NRS) Workshop and Challange: 2013 (co-chair)
ACM Sigcomm Workshop on Challenged Networks (CHANTS): 2006 (co-chair)
IEEE International Conference on Network Protocols (ICNP): 2003 (co-chair), 2006
International Workshop on Wireless Network Measurements (WiNMee): 2006 (co-chair)
IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS): 2002 (co-chair)
International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV): 2002 (co-chair), 2003 (co-chair)
IEEE Global Internet (GI) Symposium: 2001 (co-chair)
International Workshop on Networked Group Communication (NGC): 2000 (co-chair)

Program Chair

International Conference on Communication Systems and Networks (COMSNETS): 2010
IEEE International Conference on Network Protocols (ICNP): 2008 (co-chair)
IEEE Conference on Sensor and Ad Hoc Communications and Networks (SECON): 2007 (co-chair)
IFIP Networking: 2005 (co-chair)

Posters/Demonstrations Chair

ACM Sigcomm: 2012 (co-chair)

Student Travel Grants Chair

ACM Sigcomm: 2010 (co-chair)

Publicity Chair

IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS): 2004 (co-chair)

Keynote Chair

IEEE Infocom: 2005 (co-chair)

Local Arrangements Chair

Internet2 "Field of Dreams" Workshop: 2000

Tutorial Chair

ACM Multimedia: 2000
IEEE International Conference on Network Protocols (ICNP): 1999

Panel/Session Organizer

NSF ANIR PI 2003 Panel on "Core Networking Issues and Protocols for the Internet"
CCW 2001 Session on "Multicast/Peer-to-Peer Networking"
NOSSDAV 2001 Panel on "Multimedia After a Decade of Research"
NGC 2000 Panel on "Multicast Pricing"

Technical Program Committee

IEEE International Conference on Network Protocols (ICNP): 1999, 2000, 2001, 2003, 2004, 2005, 2006, 2007, 2008, 2009 (Area Chair), 2010 (Area Chair), 2011 (Area Chair), 2012 (Area Chair), 2013, 2014 (Area Chair), 2015 (Area Chair)
International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV): 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015
ACM Multimedia (MM): 2001, 2003, 2004, 2005 (short paper), 2006, 2007, 2008, 2008 (short paper), 2010, 2011, 2012, 2013, 2015
IEEE Conference on Sensor and Ad Hoc Communications and Networks (SECON): 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011 (Area Chair), 2012 (Area Chair), 2013, 2014 (Area Chair), 2015
IEEE/IFIP Network Operations and Management Symposium (NOMS): 2004, 2006, 2010
IEEE Infocom: 2004, 2005, 2006, 2008, 2009, 2010 (Area Chair), 2011 (Area Chair), 2012 (Area Chair)
IFIP Networking: 2004, 2005, 2006, 2007, 2010, 2011, 2012, 2013, 2014, 2015
ACM Workshop on Mobile Video (MoVid): 2014, 2015

ACM Student Research Competition (SRC) Grand Finals: 2014
Mobile and Social Computing for Collaborative Interactions (MSC): 2014
IEEE Conference on Communications and Network Security (CNS): 2013
IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM): 2005, 2006, 2007, 2008, 2009, 2010
ACM Sigcomm Workshop on Challenged Networks (CHANTS): 2006, 2008, 2009, 2010, 2011, 2012
IEEE International Conference on Distributed Computing Systems (ICDCS): 2006, 2010, 2011, 2012, 2013
International Workshop on Wireless Network Measurements (WiNMee): 2006, 2008, 2010
ACM Sigcomm: 2008 (poster), 2010
IEEE International Conference on Computer Communication and Networks (IC3N): 2008, 2009, 2010, 2011, 2012
International Conference on Communication Systems and Networks (COMSNETS): 2009, 2010, 2011, 2012, 2013
International Conference on Sensor Networks (SENSORNETS): 2012
International Workshop on Social and Mobile Computing for Collaborative Environments (SOMOCO): 2012
Workshop on Scenarios for Network Evaluation Studies (SCENES): 2009, 2010, 2011
ACM Multimedia Systems (MMSys): 2010, 2011, 2012, 2015
IEEE International Conference on Pervasive Computing and Communications (PerCom): 2010
IEEE Wireless Communications and Networking Conference (WCNC): 2010, 2011
ACM International Symposium on Mobility Management and Wireless Access (MobiWac): 2010, 2011
International Conference on Computing, Networking and Communications, Internet Services and Applications Symposium (ICNC-ISA): 2012, 2013
IEEE WoWMoM Workshop on Hot Topics in Mesh Networking (HotMesh): 2010, 2011, 2012, 2013
IEEE Workshop on Pervasive Group Communication (PerGroup): 2010
ACM International Conference on Next Generation Communication (CoNEXT): 2005, 2006, 2007, 2009, 2012
IEEE International Conference on Broadband Communications, Networks, and Systems (BroadNets) Wireless Communications, Networks and Systems Symposium: 2007, 2008, 2009
IEEE International Conference on Broadband Communications, Networks, and Systems (BroadNets) Internet Technologies Symposium: 2007, 2008, 2009
International Workshop on Mobile and Networking Technologies for Social Applications (MONET): 2008, 2009
Extreme Workshop on Communication-The Midnight Sun Expedition (ExtremeCom): 2009
IEEE International Workshop on Cooperation in Pervasive Environments (CoPE): 2009
International Workshop on the Network of the Future (FutureNet): 2009, 2010, 2011, 2012
IEEE International Conference on Multimedia and Exposition (ICME): 2010
SPIE Conference on Multimedia Computing and Networking (MMCN): 2004, 2008
ACM Sigcomm Workshop on the Economics of Networks, Systems, and Computation (NetEcon): 2008
IEEE International Conference on Communications (ICC): 2008

IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS): 2008
IFIP/IEEE International Symposium on Integrated Network Management (IM): 2005, 2007
Global Internet (GI) Symposium: 2001, 2002, 2004, 2006, 2007
IEEE/ACM International Conference on High Performance Computing (HiPC): 2007
ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc): 2007
IEEE Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia): 2007
IEEE/IFIP Wireless On Demand Network Systems and Services (WONS): 2007
IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS): 2001, 2002, 2003, 2004, 2005, 2006
IASTED International Conference on European Internet and Multimedia Systems and Applications (EuroIMSA): 2004, 2006
IEEE International Conference on Parallel and Distributed Systems (ICPADS): 2005, 2006
IEEE Globecom Internet Services and Enabling Technologies (IS&ET) Symposium: 2006
International Workshop on Incentive-Based Computing (IBC): 2006
IEEE International Workshop on Quality of Service (IWQoS): 2006, 2014, 2015
International Workshop on Multi-hop Ad Hoc Networks (REALMAN): 2006
IEEE Globecom Automatic Internet Symposium: 2005
ACM Sigcomm Workshop on the Economics of Peer-to-Peer Systems (P2PEcon): 2005
International Conference on Parallel Processing (ICPP): 2001, 2003, 2004
International Packet Video (PV) Workshop: 2002, 2003, 2004
IEEE International Symposium on High-Performance Distributed Computing (HPDC): 2004
ACM Sigcomm: 2004 (poster)
International Workshop on Broadband Wireless Multimedia: Algorithms, Architectures and Applications (BroadWIM): 2004
International Symposium on Innovation in Information & Communication Technology (ISIICT): 2004
Workshop on Coordinated Quality of Service in Distributed Systems (COQODS): 2004
IASTED International Conference on Networks and Communication Systems (NCS): 2004
IASTED International Conference on Communications, Internet, and Information Technology (CIIT): 2004
IASTED International Conference on Internet and Multimedia Systems and Applications (IMSA): 2003, 2004
International Workshop on Networked Group Communication (NGC): 1999, 2000, 2001, 2002, 2003
International Association for Development of the Information Society (IADIS) International Conference on the WWW/Internet: 2003
International Conference on Computer and Information Science (ICIS): 2003
Human.Society@Internet: 2003
IASTED International Conference on Communications and Computer Networks (CCN): 2002
The Content Delivery Networks (CDN) Event: 2001
IP Multicast Initiative Summit: 1998, 1999, 2000
Corporation for Education Network Initiatives in California (CENIC): 1999
Internet Bandwidth Management Summit (iBAND): 1998, 1999

B. Technical Activities

1. Working Groups

Internet2 Working Group on Multicast, Chair: 1998-2005
IEEE Communications Society Internet Technical Committee (ITC), Conference Coordinator: 2000-2004
IETF Multicast Directorate (MADDOGS), Member: 1999-2001
IASTED Technical Committee on the Web, Internet and Multimedia, Member: 2002-2005
Internet Engineering Task Force (IETF), various working groups: 1995-present

2. Meeting Support Work

Internet Engineering Task Force MBone broadcasts: 1995-2005
Conference MBone broadcasts: Sigcomm '99, and '00
Interop+Networld Network Operations Center (NOC) Team Member: 1995-1997
ACM Multimedia technical staff: 1994

C. University of California Committees

1. Department of Computer Science Committees

Public Relations: 2005-2006 (chair 2005-2006), 2009-2011 (chair 2009-2011)
Strategic Planning: 2000-2002, 2003-2006, 2009-2011
Undergraduate Advising and Affairs: 2006-2007, 2014-2015
Vice Chair: 2000-2005
Graduate Admissions: 2000-2005 (chair 2000-2005), 2011-2012
Graduate Affairs: 2000-2005 (co-chair 2000-2005)
Teaching Administration: 2000-2005
Facilities: 1997-2001 (chair 1999-2000), 2006-2007
External Relations: 1999-2002
Computer Engineering Public Relations: 2011-2012
Computer Engineering Awards: 2011-2012
Computer Engineering Administration/Recruiting: 1998-2001
Computer Engineering Lab and Computer Support: 1998-2001
Faculty Recruiting: 1999-2002
Graduate Advising: 1998-1999, 2000-2005

2. University Committees

Member, Campus Budget and Planning: 2013-2015

Faculty, Cognitive Science Program: 2006-present
Faculty, Technology Management Program (TMP): 2003-2014
Faculty, Media Arts and Technology (MAT) Program: 1998-2014
Faculty, Computer Engineering Degree Program: 1998-present
Steering Committee, Center for Information Technology and Society (CITS): 2012-present
Associate Director, Center for Information Technology and Society (CITS): 1999-2012
Member, Campus Committee on Committees: 2010-2013
Member, Campus Income and Recharge Committee: 2010-2013
Member, College of Engineering Executive Committee: 2010-2012 (chair 2011-2012), 2014-2015 (chair 2014-2015)
Member, Distinguished Teaching Award Committee: 2009, 2010, 2011
Member, Campus Classroom Design and Renovation Committee: 2003-2010
Member, ISBER Advisory Committee: 2008-2011
Member, Fulbright Campus Review Committee: 2007
Member, Faculty Outreach Grant Program Review Committee: 2007
Executive Vice Chancellor's Information Technology Fee Committee: 2005-2006
Council on Research and Instructional Resources: 2003-2006
Executive Vice Chancellor's Working Group on Graduate Diversity: 2004-2005
Member, Engineering Pavillion Planning Committee: 2003-2005
Information Technology Board: 2001-2004
Executive Committee, Center for Entrepreneurship & Engineering Management (CEEM): 2001-2004

3. System Wide Committees

UCSB Representative to the Committee on Information Technology and Telecommunications Policy (ITTP): 2003-2005
UCSB Representative to the Executive Committee, Digital Media Innovation (DiMI): 1998-2003

D. Georgia Tech Committees and Service (while a graduate student)

Graduate Student Body President: 1994-1995
Georgia Tech Executive Board: 1994-1995
Georgia Tech Alumni Association Executive Committee: 1994-1995
Dean of Students National Search Committee: 1995
Institute Strategic Planning Committee: 1994-1996

Cases in last 5 years I have been deposed or testified (I represented the underlined party):

- Two depositions in Beneficial Innovations, Inc. v. Blockdot, Inc. et al. (2:07-CV-263(TJW/CE) and 2:07-CV-555 (TJW/CE), E.D. Tex.). Finished: 10/10.
- Two depositions and trial testimony in Personal Audio, LLC v. Apple, Inc. (9:09-CV-00111-RC, E.D. Tex.). Finished: 07/11.
- Two depositions in Paltalk Holdings, Inc. v. Sony et al. (2:09-cv-274-DF-CE, E.D. Tex.). Finished: 09/11.
- A deposition and trial testimony in Certain Wireless Communication Devices, Portable Music and Data Processing Devices, Computers and Components (US ITC Inv. No. 337-TA-745) [Motorola Mobility v. Apple]. Finished: 04/12.
- Two depositions in Intermec Technologies Corp. v. Palm Inc. (07-272-SLR, D. Del.). Finished: 05/12.
- A deposition in iHance, Inc. v. Eloqua Corp. (2:11-CV-257-MSD-TEM, E.D. Va.). Finished: 06/12.
- A deposition in Apple, Inc. v. Motorola Mobility, Inc. (11-CV-178 (BBC), W.D. Wis.). Finished: 10/12.
- A deposition and trial testimony in Two-Way Media LLC v. AT&T Inc., et al. (SA-09-CA-476-OLG, W.D. Tex.). Finished: 03/13.
- Depositions in British Telecommunications PLC v. CoxCom, Inc., Cox Communications, Inc., & Cable One, Inc. (10-658-SLR, D. Del.). Finished: 01/14.
- A deposition and trial testimony in Certain Digital Media Devices, Including Televisions, Blu-Ray Disc Players, Home Theater Systems, Tablets and Mobile Phones, Components Thereof and Associated Software (ITC Inv. No. 337-TA-882) [Black Hills Media v. Samsung]. Finished 02/14.
- A deposition in Inter Partes Review of U.S. Patent No. 7,107,612 (IPR2013-00369) [Palo Alto Networks, Inc. v. Juniper Networks, Inc.]. Finished 05/14.
- A deposition and trial testimony in EON Corp Holdings, LCC. v. Landis+Gyr, Inc, et al. (6:11-CV-317-LED-JDL, E.D. Tex.). Finished 06/14.
- Depositions in Straight Path IP Group, Inc. v. Bandwidth.com, Inc., Telesphere Networks Ltd., and Vocalocity, Inc. (1:13-CV-932, E.D. Va.). Finished 06/14.
- Depositions and trial testimony in Beneficial Innovations, Inc. v. Advanced Publications, Inc. et al. (2:11-CV-229-JRG-RSP, E.D. Tex.). Finished 07/14.
- Depositions in Robocast Inc. v. Apple Inc. (11-235-RGA, D. Del.) and Robocast Inc. v. Microsoft Corp. (10-1055-RGA, D. Del.). Finished 08/14.
- A deposition in PersonalWeb Technologies, LCC v. Yahoo! Inc. (6:12-CV-658-LED, E.D. Tex.). Finished 08/14.
- Depositions and trial testimony in Personal Audio LLC v. Togi Entertainment, Inc. et al. (2:13-CV-13-JRG-RSP, E.D. Tex.). Finished 09/14.
- A deposition in Inter Partes Review of U.S. Patent Nos. 8,326,924 and 8.239,451 (CBM2014-00001 and CBM2014-00050, respectively) [American Express Co. et al. v. Metasearch Systems, LLC]. Finished 09/14.
- Depositions in Inter Partes Review of U.S. Patent Nos. 6,044,062 (IPR2013-00482) and 6,249,516 (IPR2014-00147) [ABB Technology LTD v. IPCO, LLC]. Finished 10/14.

- A deposition in Inter Partes Review of U.S. Patent No. 5,995,091 (IPR2014-00153 and IPR2014-00154) [Adobe Systems Inc & Level3 Communications, LLC v. Afluo, LLC]. Finished 10/14.
- Depositions in Inter Partes Review of U.S. Patent Nos 8,145,268; 8,224,381; 8,135,398; 7,899,492; 8,050,711; and 8,712,471 (IPR2013-00569, IPR2013-00570, IPR2013-00571, IPR2013-00572, IPR2013-00573 and IPR2015-00054, respectively) [Samsung Electronics Co., LTD v. Virginia Innovation Sciences, Inc.]. Finished 11/14.
- A deposition in Black Hills Media, LLC v. Sonos, Inc. (14-cv-00486-SJC-PJWx, C.D. Cal.). Finished 02/15.
- Markman testimony in Personal Audio, LLC v. Apollo Brands et al. (1:14-CV-8-RC, E.D. Tex.). Finished 06/15.
- Depositions in Inter Partes Review of U.S. Patent Nos. 8,028,323; 8,230,099; 8,214,873; 6,108,686; 7,835,689; and 7,917,082 (IPR2014-00709, IPR2014-00711, IPR2014-00723, IPR2014-00717, IPR2014-00718, and IPR2014-00721, respectively) [Samsung Electronics Co., LTD v. Black Hills Media, LLC]. Finished 06/15.
- A deposition in Inter Partes Review of U.S. Patent No. 7,548,875 (IPR2014-01236) [MindGeek et al. v. Skky, Inc.]. Finished 06/15.
- A deposition in Inter Partes Review of U.S. Patent Nos. 7,468,661 (IPR2014-00751) [Hart Communication Foundation v. SIPCO, LLC]. Finished 07/15.
- Depositions in Inter Partes Review of U.S. Patent No. 6,754,195 (IPR2014-00552 and IPR2014-00553) [Marvell Semiconductor, Inc. v. Intellectual Ventures I LLC]. Finished 07/15.
- A deposition and trial testimony in Certain Network Devices, Related Software and Components Thereof (US ITC Inv. No. 337-TA-944) [Cisco v. Arista]. Finished 09/15.
- A deposition in Sprint Communications Company LP v. Time Warner Cable, Inc. (11-2686-JWL, D. Kan.);
- Depositions in Certain Network Devices, Related Software and Components Thereof (II) (US ITC Inv. No. 337-TA-945) [Cisco v. Arista];

APPENDIX C

HTML

Living Standard — Last Updated 30 October 2015

Table of Contents — 2 Common infrastructure →

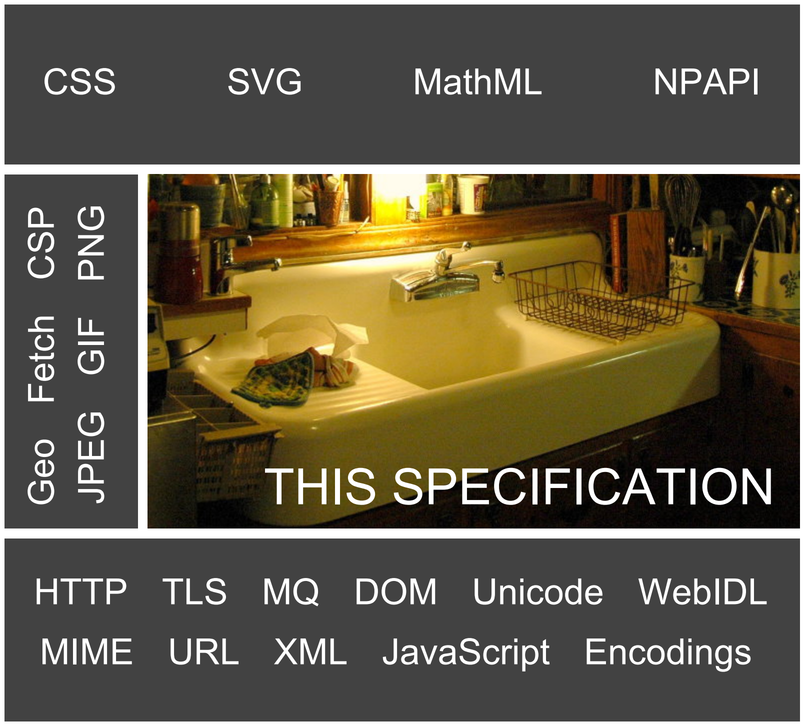
File an issue about the selected text

1 Introduction

Spec bugs: 23036

1.1 Where does this specification fit?

This specification defines a big part of the Web platform, in lots of detail. Its place in the Web platform specification stack relative to other specifications can be best summed up as follows:



1.2 Is this HTML5?

This section is non-normative.

In short: Yes.

In more length: The term "HTML5" is widely used as a buzzword to refer to modern Web technologies, many of which (though by no means all) are developed at the WHATWG. This document is one such; others are available from the WHATWG specification index.

Note *Although we have asked them to stop doing so, the W3C also republishes some parts of this specification as separate documents.*

1.3 Background

This section is non-normative.

HTML is the World Wide Web's core markup language. Originally, HTML was primarily designed as a language for semantically describing scientific documents. Its general design, however, has enabled it to be adapted, over the subsequent years, to describe a number of other types of documents and even applications.

File an issue about the selected text

1.4 Audience

This section is non-normative.

This specification is intended for authors of documents and scripts that use the features defined in this specification, implementors of tools that operate on pages that use the features defined in this specification, and individuals wishing to establish the correctness of documents or implementations with respect to the requirements of this specification.

This document is probably not suited to readers who do not already have at least a passing familiarity with Web technologies, as in places it sacrifices clarity for precision, and brevity for completeness. More approachable tutorials and authoring guides can provide a gentler introduction to the topic.

In particular, familiarity with the basics of DOM is necessary for a complete understanding of some of the more technical parts of this specification. An understanding of Web IDL, HTTP, XML, Unicode, character encodings, JavaScript, and CSS will also be helpful in places but is not essential.

1.5 Scope

This section is non-normative.

This specification is limited to providing a semantic-level markup language and associated semantic-level scripting APIs for authoring accessible pages on the Web ranging from static documents to dynamic applications.

The scope of this specification does not include providing mechanisms for media-specific customization of presentation (although default rendering rules for Web browsers are included at the end of this specification, and several mechanisms for hooking into CSS are provided as part of the language).

The scope of this specification is not to describe an entire operating system. In particular, hardware configuration software, image manipulation tools, and applications that users would be expected to use with high-end workstations on a daily basis are out of scope. In terms of applications, this specification is targeted specifically at applications that would be expected to be used by users on an occasional basis, or regularly but from disparate locations, with low CPU requirements. Examples of such applications include online purchasing systems, searching systems, games (especially multiplayer online games), public telephone books or address books, communications software (e-mail clients, instant messaging clients, discussion software), document editing software, etc.

1.6 History

This section is non-normative.

For its first five years (1990-1995), HTML went through a number of revisions and experienced a number of extensions, primarily hosted first at CERN, and then at the IETF.

With the creation of the W3C, HTML's development changed venue again. A first abortive attempt at extending HTML in 1995 known as HTML 3.0 then made way to a more pragmatic approach known as HTML 3.2, which was completed in 1997. HTML4 quickly followed later that same year.

The following year, the W3C membership decided to stop evolving HTML and instead begin work on an XML-based equivalent, called XHTML. This effort started with a reformulation of HTML4 in XML, known as XHTML 1.0, which added no new features except the new serialisation, and which was completed in 2000. After XHTML 1.0, the W3C's focus turned to making it easier for other working groups to extend XHTML, under the banner of XHTML Modularization. In parallel with this, the W3C also worked on a new language that was not compatible with the earlier HTML and XHTML languages, calling it XHTML2.

Around the time that HTML's evolution was stopped in 1998, parts of the API for HTML developed by browser vendors were specified and published under the name DOM Level 1 (in 1998) and DOM Level 2 Core and DOM Level 2 HTML (starting in 2000 and

File an issue about the selected text

culminating in 2003). These efforts then petered out, with some DOM Level 3 specifications published in 2004 but the working group being closed before all the Level 3 drafts were completed.

In 2003, the publication of XForms, a technology which was positioned as the next generation of Web forms, sparked a renewed interest in evolving HTML itself, rather than finding replacements for it. This interest was borne from the realization that XML's deployment as a Web technology was limited to entirely new technologies (like RSS and later Atom), rather than as a replacement for existing deployed technologies (like HTML).

A proof of concept to show that it was possible to extend HTML4's forms to provide many of the features that XForms 1.0 introduced, without requiring browsers to implement rendering engines that were incompatible with existing HTML Web pages, was the first result of this renewed interest. At this early stage, while the draft was already publicly available, and input was already being solicited from all sources, the specification was only under Opera Software's copyright.

The idea that HTML's evolution should be reopened was tested at a W3C workshop in 2004, where some of the principles that underlie the HTML5 work (described below), as well as the aforementioned early draft proposal covering just forms-related features, were presented to the W3C jointly by Mozilla and Opera. The proposal was rejected on the grounds that the proposal conflicted with the previously chosen direction for the Web's evolution; the W3C staff and membership voted to continue developing XML-based replacements instead.

Shortly thereafter, Apple, Mozilla, and Opera jointly announced their intent to continue working on the effort under the umbrella of a new venue called the WHATWG. A public mailing list was created, and the draft was moved to the WHATWG site. The copyright was subsequently amended to be jointly owned by all three vendors, and to allow reuse of the specification.

The WHATWG was based on several core principles, in particular that technologies need to be backwards compatible, that specifications and implementations need to match even if this means changing the specification rather than the implementations, and that specifications need to be detailed enough that implementations can achieve complete interoperability without reverse-engineering each other.

The latter requirement in particular required that the scope of the HTML5 specification include what had previously been specified in three separate documents: HTML4, XHTML1, and DOM2 HTML. It also meant including significantly more detail than had previously been considered the norm.

In 2006, the W3C indicated an interest to participate in the development of HTML5 after all, and in 2007 formed a working group chartered to work with the WHATWG on the development of the HTML5 specification. Apple, Mozilla, and Opera allowed the W3C to publish the specification under the W3C copyright, while keeping a version with the less restrictive license on the WHATWG site.

For a number of years, both groups then worked together. In 2011, however, the groups came to the conclusion that they had different goals: the W3C wanted to publish a "finished" version of "HTML5", while the WHATWG wanted to continue working on a Living Standard for HTML, continuously maintaining the specification rather than freezing it in a state with known problems, and adding new features as needed to evolve the platform.

Since then, the WHATWG has been working on this specification (amongst others), and the W3C has been copying fixes made by the WHATWG into their fork of the document (which also has other changes).

1.7 Design notes

This section is non-normative.

It must be admitted that many aspects of HTML appear at first glance to be nonsensical and inconsistent.

HTML, its supporting DOM APIs, as well as many of its supporting technologies, have been developed over a period of several decades by a wide array of people with different priorities who, in many cases, did not know of each other's existence.

Features have thus arisen from many sources, and have not always been designed in especially consistent ways. Furthermore, because of the unique characteristics of the Web, implementation bugs have often become de-facto, and now de-jure, standards, as content is often unintentionally written in ways that rely on them before they can be fixed.

File an issue about the selected text

Despite all this, efforts have been made to adhere to certain design goals. These are described in the next few subsections.

1.7.1 Serialisability of script execution

This section is non-normative.

To avoid exposing Web authors to the complexities of multithreading, the HTML and DOM APIs are designed such that no script can ever detect the simultaneous execution of other scripts. Even with workers, the intent is that the behaviour of implementations can be thought of as completely serialising the execution of all scripts in all browsing contexts.

Note *The `navigator.yieldForStorageUpdates()` method, in this model, is equivalent to allowing other scripts to run while the calling script is blocked.*

1.7.2 Compliance with other specifications

This section is non-normative.

This specification interacts with and relies on a wide variety of other specifications. In certain circumstances, unfortunately, conflicting needs have led to this specification violating the requirements of these other specifications. Whenever this has occurred, the transgressions have each been noted as a "**willful violation**", and the reason for the violation has been noted.

1.7.3 Extensibility

This section is non-normative.

HTML has a wide array of extensibility mechanisms that can be used for adding semantics in a safe manner:

- Authors can use the `class` attribute to extend elements, effectively creating their own elements, while using the most applicable existing "real" HTML element, so that browsers and other tools that don't know of the extension can still support it somewhat well. This is the tack used by microformats, for example.
- Authors can include data for inline client-side scripts or server-side site-wide scripts to process using the `data-*=""` attributes. These are guaranteed to never be touched by browsers, and allow scripts to include data on HTML elements that scripts can then look for and process.
- Authors can use the `<meta name="" content="">` mechanism to include page-wide metadata by registering extensions to the predefined set of metadata names.
- Authors can use the `rel=""` mechanism to annotate links with specific meanings by registering extensions to the predefined set of link types. This is also used by microformats.
- Authors can embed raw data using the `<script type="">` mechanism with a custom type, for further handling by inline or server-side scripts.
- Authors can create plugins and invoke them using the `embed` element. This is how Flash works.
- Authors can extend APIs using the JavaScript prototyping mechanism. This is widely used by script libraries, for instance.
- Authors can use the microdata feature (the `itemscope=""` and `itemprop=""` attributes) to embed nested name-value pairs of data to be shared with other applications and sites.

File an issue about the selected text

1.8 HTML vs XHTML

This section is non-normative.

This specification defines an abstract language for describing documents and applications, and some APIs for interacting with in-memory representations of resources that use this language.

The in-memory representation is known as "DOM HTML", or "the DOM" for short.

There are various concrete syntaxes that can be used to transmit resources that use this abstract language, two of which are defined in this specification.

The first such concrete syntax is the HTML syntax. This is the format suggested for most authors. It is compatible with most legacy Web browsers. If a document is transmitted with the `text/html` MIME type, then it will be processed as an HTML document by Web browsers. This specification defines the latest HTML syntax, known simply as "HTML".

The second concrete syntax is the XHTML syntax, which is an application of XML. When a document is transmitted with an XML MIME type, such as `application/xhtml+xml`, then it is treated as an XML document by Web browsers, to be parsed by an XML processor. Authors are reminded that the processing for XML and HTML differs; in particular, even minor syntax errors will prevent a document labeled as XML from being rendered fully, whereas they would be ignored in the HTML syntax. This specification defines the latest XHTML syntax, known simply as "XHTML".

The DOM, the HTML syntax, and the XHTML syntax cannot all represent the same content. For example, namespaces cannot be represented using the HTML syntax, but they are supported in the DOM and in the XHTML syntax. Similarly, documents that use the `noscript` feature can be represented using the HTML syntax, but cannot be represented with the DOM or in the XHTML syntax. Comments that contain the string "`-->`" can only be represented in the DOM, not in the HTML and XHTML syntaxes.

1.9 Structure of this specification

Spec bugs: 26942

This section is non-normative.

This specification is divided into the following major sections:

Introduction

Non-normative materials providing a context for the HTML standard.

Common infrastructure

The conformance classes, algorithms, definitions, and the common underpinnings of the rest of the specification.

Semantics, structure, and APIs of HTML documents

Documents are built from elements. These elements form a tree using the DOM. This section defines the features of this DOM, as well as introducing the features common to all elements, and the concepts used in defining elements.

The elements of HTML

Each element has a predefined meaning, which is explained in this section. Rules for authors on how to use the element, along with user agent requirements for how to handle each element, are also given. This includes large signature features of HTML such as video playback and subtitles, form controls and form submission, and a 2D graphics API known as the HTML canvas.

Microdata

This specification introduces a mechanism for adding machine-readable annotations to documents, so that tools can extract trees of name-value pairs from the document. This section describes this mechanism and some algorithms that can be used to convert HTML documents into other formats. This section also defines some sample Microdata vocabularies for contact information, calendar events, and licensing works.

User interaction

HTML documents can provide a number of mechanisms for users to interact with and modify content, which are described in this section, such as how focus works, and drag-and-drop.

File an issue about the selected text

Loading Web pages
HTML documents do not exist in a vacuum — this section defines many of the features that affect environments that deal with multiple pages, such as Web browsers and offline caching of Web applications.

Web application APIs
This section introduces basic features for scripting of applications in HTML.

Web workers
This section defines an API for background threads in JavaScript.

The communication APIs
This section describes some mechanisms that applications written in HTML can use to communicate with other applications from different domains running on the same client. It also introduces a server-push event stream mechanism known as Server Sent Events or `EventSource`, and a two-way full-duplex socket protocol for scripts known as Web Sockets.

Web storage
This section defines a client-side storage mechanism based on name-value pairs.

The HTML syntax
The XHTML syntax
All of these features would be for naught if they couldn't be represented in a serialised form and sent to other people, and so these sections define the syntaxes of HTML and XHTML, along with rules for how to parse content using those syntaxes.

Rendering
This section defines the default rendering rules for Web browsers.

There are also some appendices, listing obsolete features and IANA considerations, and several indices.

1.9.1 How to read this specification

This specification should be read like all other specifications. First, it should be read cover-to-cover, multiple times. Then, it should be read backwards at least once. Then it should be read by picking random sections from the contents list and following all the cross-references.

As described in the conformance requirements section below, this specification describes conformance criteria for a variety of conformance classes. In particular, there are conformance requirements that apply to *producers*, for example authors and the documents they create, and there are conformance requirements that apply to *consumers*, for example Web browsers. They can be distinguished by what they are requiring: a requirement on a producer states what is allowed, while a requirement on a consumer states how software is to act.

Example For example, "the `foo` attribute's value must be a valid integer" is a requirement on producers, as it lays out the allowed values; in contrast, the requirement "the `foo` attribute's value must be parsed using the rules for parsing integers" is a requirement on consumers, as it describes how to process the content.

Requirements on producers have no bearing whatsoever on consumers.

Example Continuing the above example, a requirement stating that a particular attribute's value is constrained to being a valid integer emphatically does *not* imply anything about the requirements on consumers. It might be that the consumers are in fact required to treat the attribute as an opaque string, completely unaffected by whether the value conforms to the requirements or not. It might be (as in the previous example) that the consumers are required to parse the value using specific rules that define how invalid (non-numeric in this case) values are to be processed.

1.9.2 Typographic conventions

This is a definition, requirement, or explanation.

File an issue about the selected text

Note *This is a note.*

Example This is an example.

This is an open issue.

⚠Warning! ***This is a warning.***

IDL
```
interface Example {
  // this is an IDL definition
};
```

Note

***variable* = *object* . method([*optionalArgument*])**

This is a note to authors describing the usage of an interface.

CSS
```
/* this is a CSS fragment */
```

The defining instance of a term is marked up like **this**. Uses of that term are marked up like this or like *this*.

The defining instance of an element, attribute, or API is marked up like **`this`**. References to that element, attribute, or API are marked up like `this`.

Other code fragments are marked up `like this`.

Byte sequences with bytes in the range 0x00 to 0x7F, inclusive, are marked up like `` `this` ``.

Variables are marked up like *this*.

In an algorithm, steps in synchronous sections are marked with ⌛.

In some cases, requirements are given in the form of lists with conditions and corresponding requirements. In such cases, the requirements that apply to a condition are always the first set of requirements that follow the condition, even in the case of there being multiple sets of conditions for those requirements. Such cases are presented as follows:

↪ **This is a condition**
↪ **This is another condition**

This is the requirement that applies to the conditions above.

↪ **This is a third condition**

This is the requirement that applies to the third condition.

1.10 Privacy concerns

This section is non-normative.

Some features of HTML trade user convenience for a measure of user privacy.

In general, due to the Internet's architecture, a user can be distinguished from another by the user's IP address. IP addresses do not perfectly match to a user; as a user moves from device to device, or from network to network, their IP address will change; similarly, NAT routing, proxy servers, and shared computers enable packets that appear to all come from a single IP address to actually map to

File an issue about the selected text

multiple users. Technologies such as onion routing can be used to further anonymise requests so that requests from a single user at one node on the Internet appear to come from many disparate parts of the network.

However, the IP address used for a user's requests is not the only mechanism by which a user's requests could be related to each other. Cookies, for example, are designed specifically to enable this, and are the basis of most of the Web's session features that enable you to log into a site with which you have an account.

There are other mechanisms that are more subtle. Certain characteristics of a user's system can be used to distinguish groups of users from each other; by collecting enough such information, an individual user's browser's "digital fingerprint" can be computed, which can be as good, if not better, as an IP address in ascertaining which requests are from the same user.

Grouping requests in this manner, especially across multiple sites, can be used for both benign (and even arguably positive) purposes, as well as for malevolent purposes. An example of a reasonably benign purpose would be determining whether a particular person seems to prefer sites with dog illustrations as opposed to sites with cat illustrations (based on how often they visit the sites in question) and then automatically using the preferred illustrations on subsequent visits to participating sites. Malevolent purposes, however, could include governments combining information such as the person's home address (determined from the addresses they use when getting driving directions on one site) with their apparent political affiliations (determined by examining the forum sites that they participate in) to determine whether the person should be prevented from voting in an election.

Since the malevolent purposes can be remarkably evil, user agent implementors are encouraged to consider how to provide their users with tools to minimise leaking information that could be used to fingerprint a user.

Unfortunately, as the first paragraph in this section implies, sometimes there is great benefit to be derived from exposing the very information that can also be used for fingerprinting purposes, so it's not as easy as simply blocking all possible leaks. For instance, the ability to log into a site to post under a specific identity requires that the user's requests be identifiable as all being from the same user, more or less by definition. More subtly, though, information such as how wide text is, which is necessary for many effects that involve drawing text onto a canvas (e.g. any effect that involves drawing a border around the text) also leaks information that can be used to group a user's requests. (In this case, by potentially exposing, via a brute force search, which fonts a user has installed, information which can vary considerably from user to user.)

Features in this specification which can be **used to fingerprint the user** are marked as this paragraph is.

Other features in the platform can be used for the same purpose, though, including, though not limited to:

- The exact list of which features a user agents supports.
- The maximum allowed stack depth for recursion in script.
- Features that describe the user's environment, like Media Queries and the `Screen` object. [MQ] [CSSOMVIEW]
- The user's time zone.

1.10.1 Cross-site communication

The `postMessage()` API provides a mechanism by which two sites can communicate directly. At first glance, this might appear to open a new way by which the problems described above can occur. However, in practice, multiple mechanisms exist by which two sites can communicate that predate this API: a site embedding another can send data via an `iframe` element's dimensions; a site can use a cross-site image request with a unique identifier known to the server to initiate a server-side data exchange; or indeed the fingerprinting techniques described above can be used by two sites to uniquely identify a visitor such that information can then be exchanged on the server side.

Fundamentally, users that do not trust a site to treat their information with respect have to avoid visiting that site at all.

1.11 A quick introduction to HTML

This section is non-normative.

A basic HTML document looks like this:

```
<!DOCTYPE html>
<html>
 <head>
  <title>Sample page</title>
 </head>
 <body>
  <h1>Sample page</h1>
  <p>This is a <a href="demo.html">simple</a> sample.</p>
  <!-- this is a comment -->
 </body>
</html>
```

HTML documents consist of a tree of elements and text. Each element is denoted in the source by a start tag, such as "`<body>`", and an end tag, such as "`</body>`". (Certain start tags and end tags can in certain cases be omitted and are implied by other tags.)

Tags have to be nested such that elements are all completely within each other, without overlapping:

```
<p>This is <em>very <strong>wrong</em>!</strong></p>
```

```
<p>This <em>is <strong>correct</strong>.</em></p>
```

This specification defines a set of elements that can be used in HTML, along with rules about the ways in which the elements can be nested.

Elements can have attributes, which control how the elements work. In the example below, there is a hyperlink, formed using the `a` element and its `href` attribute:

```
<a href="demo.html">simple</a>
```

Attributes are placed inside the start tag, and consist of a name and a value, separated by an "`=`" character. The attribute value can remain unquoted if it doesn't contain space characters or any of `"` `'` `` ` `` `=` `<` or `>`. Otherwise, it has to be quoted using either single or double quotes. The value, along with the "`=`" character, can be omitted altogether if the value is the empty string.

```
<!-- empty attributes -->
<input name=address disabled>
<input name=address disabled="">

<!-- attributes with a value -->
<input name=address maxlength=200>
<input name=address maxlength='200'>
<input name=address maxlength="200">
```

HTML user agents (e.g. Web browsers) then *parse* this markup, turning it into a DOM (Document Object Model) tree. A DOM tree is an in-memory representation of a document.

DOM trees contain several kinds of nodes, in particular a `DocumentType` node, `Element` nodes, `Text` nodes, `Comment` nodes, and in some cases `ProcessingInstruction` nodes.

The markup snippet at the top of this section would be turned into the following DOM tree:

```
├DOCTYPE: html
└html
 ├head
 │├#text: ⏎␣␣
 │└
```

File an issue about the selected text

```
│ ┌title
│ │ └#text: Sample page
│ └#text: ⏎␣
├#text: ⏎␣
└body
  ├#text: ⏎␣␣
  ├h1
  │ └#text: Sample page
  ├#text: ⏎␣␣
  ├p
  │ ├#text: This is a
  │ ├a href="demo.html"
  │ │ └#text: simple
  │ └#text: sample.
  ├#text: ⏎␣␣
  ├#comment: this is a comment
  └#text: ⏎␣⏎
```

The root element of this tree is the `html` element, which is the element always found at the root of HTML documents. It contains two elements, `head` and `body`, as well as a `Text` node between them.

There are many more `Text` nodes in the DOM tree than one would initially expect, because the source contains a number of spaces (represented here by "␣") and line breaks ("⏎") that all end up as `Text` nodes in the DOM. However, for historical reasons not all of the spaces and line breaks in the original markup appear in the DOM. In particular, all the whitespace before `head` start tag ends up being dropped silently, and all the whitespace after the `body` end tag ends up placed at the end of the `body`.

The `head` element contains a `title` element, which itself contains a `Text` node with the text "Sample page". Similarly, the `body` element contains an `h1` element, a `p` element, and a comment.

This DOM tree can be manipulated from scripts in the page. Scripts (typically in JavaScript) are small programs that can be embedded using the `script` element or using event handler content attributes. For example, here is a form with a script that sets the value of the form's `output` element to say "Hello World":

```
<form name="main">
 Result: <output name="result"></output>
 <script>
  document.forms.main.elements.result.value = 'Hello World';
 </script>
</form>
```

Each element in the DOM tree is represented by an object, and these objects have APIs so that they can be manipulated. For instance, a link (e.g. the `a` element in the tree above) can have its "`href`" attribute changed in several ways:

```
var a = document.links[0]; // obtain the first link in the document
a.href = 'sample.html'; // change the destination URL of the link
a.protocol = 'https'; // change just the scheme part of the URL
a.setAttribute('href', 'http://example.com/'); // change the content attribute directly
```

Since DOM trees are used as the way to represent HTML documents when they are processed and presented by implementations (especially interactive implementations like Web browsers), this specification is mostly phrased in terms of DOM trees, instead of the markup described above.

File an issue about the selected text

HTML documents represent a media-independent description of interactive content. HTML documents might be rendered to a screen, or through a speech synthesiser, or on a braille display. To influence exactly how such rendering takes place, authors can use a styling language such as CSS.

In the following example, the page has been made yellow-on-blue using CSS.

```
<!DOCTYPE html>
<html>
 <head>
  <title>Sample styled page</title>
  <style>
   body { background: navy; color: yellow; }
  </style>
 </head>
 <body>
  <h1>Sample styled page</h1>
  <p>This page is just a demo.</p>
 </body>
</html>
```

For more details on how to use HTML, authors are encouraged to consult tutorials and guides. Some of the examples included in this specification might also be of use, but the novice author is cautioned that this specification, by necessity, defines the language with a level of detail that might be difficult to understand at first.

1.11.1 Writing secure applications with HTML

This section is non-normative.

When HTML is used to create interactive sites, care needs to be taken to avoid introducing vulnerabilities through which attackers can compromise the integrity of the site itself or of the site's users.

A comprehensive study of this matter is beyond the scope of this document, and authors are strongly encouraged to study the matter in more detail. However, this section attempts to provide a quick introduction to some common pitfalls in HTML application development.

The security model of the Web is based on the concept of "origins", and correspondingly many of the potential attacks on the Web involve cross-origin actions. [ORIGIN]

Not validating user input
Cross-site scripting (XSS)
SQL injection

When accepting untrusted input, e.g. user-generated content such as text comments, values in URL parameters, messages from third-party sites, etc, it is imperative that the data be validated before use, and properly escaped when displayed. Failing to do this can allow a hostile user to perform a variety of attacks, ranging from the potentially benign, such as providing bogus user information like a negative age, to the serious, such as running scripts every time a user looks at a page that includes the information, potentially propagating the attack in the process, to the catastrophic, such as deleting all data in the server.

When writing filters to validate user input, it is imperative that filters always be safelist-based, allowing known-safe constructs and disallowing all other input. Blocklist-based filters that disallow known-bad inputs and allow everything else are not secure, as not everything that is bad is yet known (for example, because it might be invented in the future).

Example For example, suppose a page looked at its URL's query string to determine what to display, and the site then redirected the user to that page to display a message, as in:

```
<ul>
 <li><a href="message.cgi?say=Hello">Say Hello</a>
```

File an issue about the selected text

```
 <li><a href="message.cgi?say=Welcome">Say Welcome</a>
 <li><a href="message.cgi?say=Kittens">Say Kittens</a>
</ul>
```

If the message was just displayed to the user without escaping, a hostile attacker could then craft a URL that contained a script element:

```
http://example.com/message.cgi?
say=%3Cscript%3Ealert%28%27Oh%20no%21%27%29%3C/script%3E
```

If the attacker then convinced a victim user to visit this page, a script of the attacker's choosing would run on the page. Such a script could do any number of hostile actions, limited only by what the site offers: if the site is an e-commerce shop, for instance, such a script could cause the user to unknowingly make arbitrarily many unwanted purchases.

This is called a cross-site scripting attack.

There are many constructs that can be used to try to trick a site into executing code. Here are some that authors are encouraged to consider when writing safelist filters:

- When allowing harmless-seeming elements like `img`, it is important to safelist any provided attributes as well. If one allowed all attributes then an attacker could, for instance, use the `onload` attribute to run arbitrary script.
- When allowing URLs to be provided (e.g. for links), the scheme of each URL also needs to be explicitly safelisted, as there are many schemes that can be abused. The most prominent example is "`javascript:`", but user agents can implement (and indeed, have historically implemented) others.
- Allowing a `base` element to be inserted means any `script` elements in the page with relative links can be hijacked, and similarly that any form submissions can get redirected to a hostile site.

Cross-site request forgery (CSRF)

If a site allows a user to make form submissions with user-specific side-effects, for example posting messages on a forum under the user's name, making purchases, or applying for a passport, it is important to verify that the request was made by the user intentionally, rather than by another site tricking the user into making the request unknowingly.

This problem exists because HTML forms can be submitted to other origins.

Sites can prevent such attacks by populating forms with user-specific hidden tokens, or by checking `` `Origin` `` headers on all requests.

Clickjacking

A page that provides users with an interface to perform actions that the user might not wish to perform needs to be designed so as to avoid the possibility that users can be tricked into activating the interface.

One way that a user could be so tricked is if a hostile site places the victim site in a small `iframe` and then convinces the user to click, for instance by having the user play a reaction game. Once the user is playing the game, the hostile site can quickly position the iframe under the mouse cursor just as the user is about to click, thus tricking the user into clicking the victim site's interface.

To avoid this, sites that do not expect to be used in frames are encouraged to only enable their interface if they detect that they are not in a frame (e.g. by comparing the `window` object to the value of the `top` attribute).

1.11.2 Common pitfalls to avoid when using the scripting APIs

This section is non-normative.

Scripts in HTML have "run-to-completion" semantics, meaning that the browser will generally run the script uninterrupted before doing anything else, such as firing further events or continuing to parse the document.

File an issue about the selected text

On the other hand, parsing of HTML files happens incrementally, meaning that the parser can pause at any point to let scripts run. This is generally a good thing, but it does mean that authors need to be careful to avoid hooking event handlers after the events could have possibly fired.

There are two techniques for doing this reliably: use event handler content attributes, or create the element and add the event handlers in the same script. The latter is safe because, as mentioned earlier, scripts are run to completion before further events can fire.

Example One way this could manifest itself is with `img` elements and the `load` event. The event could fire as soon as the element has been parsed, especially if the image has already been cached (which is common).

Here, the author uses the `onload` handler on an `img` element to catch the `load` event:

```
<img src="games.png" alt="Games" onload="gamesLogoHasLoaded(event)">
```

If the element is being added by script, then so long as the event handlers are added in the same script, the event will still not be missed:

```
<script>
 var img = new Image();
 img.src = 'games.png';
 img.alt = 'Games';
 img.onload = gamesLogoHasLoaded;
 // img.addEventListener('load', gamesLogoHasLoaded, false); // would work also
</script>
```

However, if the author first created the `img` element and then in a separate script added the event listeners, there's a chance that the `load` event would be fired in between, leading it to be missed:

```
<!-- Do not use this style, it has a race condition! -->
 <img id="games" src="games.png" alt="Games">
 <!-- the 'load' event might fire here while the parser is taking a
      break, in which case you will not see it! -->
 <script>
  var img = document.getElementById('games');
  img.onload = gamesLogoHasLoaded; // might never fire!
 </script>
```

1.11.3 How to catch mistakes when writing HTML: validators and conformance checkers

This section is non-normative.

Authors are encouraged to make use of conformance checkers (also known as *validators*) to catch common mistakes. The WHATWG maintains a list of such tools at: https://validator.whatwg.org/

1.12 Conformance requirements for authors

This section is non-normative.

Unlike previous versions of the HTML specification, this specification defines in some detail the required processing for invalid documents as well as valid documents.

However, even though the processing of invalid content is in most cases well-defined, conformance requirements for documents are still important: in practice, interoperability (the situation in which all implementations process particular content in a reliable and

File an issue about the selected text

identical or equivalent way) is not the only goal of document conformance requirements. This section details some of the more common reasons for still distinguishing between a conforming document and one with errors.

1.12.1 Presentational markup

This section is non-normative.

The majority of presentational features from previous versions of HTML are no longer allowed. Presentational markup in general has been found to have a number of problems:

The use of presentational elements leads to poorer accessibility

While it is possible to use presentational markup in a way that provides users of assistive technologies (ATs) with an acceptable experience (e.g. using ARIA), doing so is significantly more difficult than doing so when using semantically-appropriate markup. Furthermore, even using such techniques doesn't help make pages accessible for non-AT non-graphical users, such as users of text-mode browsers.

Using media-independent markup, on the other hand, provides an easy way for documents to be authored in such a way that they work for more users (e.g. text browsers).

Higher cost of maintenance

It is significantly easier to maintain a site written in such a way that the markup is style-independent. For example, changing the colour of a site that uses `<font color="">` throughout requires changes across the entire site, whereas a similar change to a site based on CSS can be done by changing a single file.

Larger document sizes

Presentational markup tends to be much more redundant, and thus results in larger document sizes.

For those reasons, presentational markup has been removed from HTML in this version. This change should not come as a surprise; HTML4 deprecated presentational markup many years ago and provided a mode (HTML4 Transitional) to help authors move away from presentational markup; later, XHTML 1.1 went further and obsoleted those features altogether.

The only remaining presentational markup features in HTML are the `style` attribute and the `style` element. Use of the `style` attribute is somewhat discouraged in production environments, but it can be useful for rapid prototyping (where its rules can be directly moved into a separate style sheet later) and for providing specific styles in unusual cases where a separate style sheet would be inconvenient. Similarly, the `style` element can be useful in syndication or for page-specific styles, but in general an external style sheet is likely to be more convenient when the styles apply to multiple pages.

It is also worth noting that some elements that were previously presentational have been redefined in this specification to be media-independent: `b`, `i`, `hr`, `s`, `small`, and `u`.

1.12.2 Syntax errors

This section is non-normative.

The syntax of HTML is constrained to avoid a wide variety of problems.

Unintuitive error-handling behaviour

Certain invalid syntax constructs, when parsed, result in DOM trees that are highly unintuitive.

Example

For example, the following markup fragment results in a DOM with an `hr` element that is an *earlier* sibling of the corresponding `table` element:

```
<table><hr>...
```

File an issue about the selected text

Errors with optional error recovery

To allow user agents to be used in controlled environments without having to implement the more bizarre and convoluted error handling rules, user agents are permitted to fail whenever encountering a parse error.

Errors where the error-handling behaviour is not compatible with streaming user agents

Some error-handling behaviour, such as the behaviour for the `<table><hr>...` example mentioned above, are incompatible with streaming user agents (user agents that process HTML files in one pass, without storing state). To avoid interoperability problems with such user agents, any syntax resulting in such behaviour is considered invalid.

Errors that can result in infoset coercion

When a user agent based on XML is connected to an HTML parser, it is possible that certain invariants that XML enforces, such as comments never containing two consecutive hyphens, will be violated by an HTML file. Handling this can require that the parser coerce the HTML DOM into an XML-compatible infoset. Most syntax constructs that require such handling are considered invalid.

Errors that result in disproportionally poor performance

Certain syntax constructs can result in disproportionally poor performance. To discourage the use of such constructs, they are typically made non-conforming.

Example

For example, the following markup results in poor performance, since all the unclosed `i` elements have to be reconstructed in each paragraph, resulting in progressively more elements in each paragraph:

```
<p><i>He dreamt.
<p><i>He dreamt that he ate breakfast.
<p><i>Then lunch.
<p><i>And finally dinner.
```

The resulting DOM for this fragment would be:

```
├p
│└i
│ └#text: He dreamt.
├p
│└i
│ └i
│  └#text: He dreamt that he ate breakfast.
├p
│└i
│ └i
│  └i
│   └#text: Then lunch.
└p
 └i
  └i
   └i
    └i
     └#text: And finally dinner.
```

Errors involving fragile syntax constructs

There are syntax constructs that, for historical reasons, are relatively fragile. To help reduce the number of users who accidentally run into such problems, they are made non-conforming.

Example

For example, the parsing of certain named character references in attributes happens even with the closing semicolon being omitted. It is safe to include an ampersand followed by letters that do not form a named character reference, but if the letters are changed to a string that *does* form a named character reference, they will be interpreted as that character instead.

In this fragment, the attribute's value is "`?bill&ted`":

File an issue about the selected text

```
<a href="?bill&ted">Bill and Ted</a>
```

In the following fragment, however, the attribute's value is actually "`?art©`", *not* the intended "`?art&copy`", because even without the final semicolon, "`&copy`" is handled the same as "`&copy;`" and thus gets interpreted as "`©`":

```
<a href="?art&copy">Art and Copy</a>
```

To avoid this problem, all named character references are required to end with a semicolon, and uses of named character references without a semicolon are flagged as errors.

Thus, the correct way to express the above cases is as follows:

```
<a href="?bill&ted">Bill and Ted</a> <!-- &ted is ok, since it's not a named
character reference -->
```

```
<a href="?art&amp;copy">Art and Copy</a> <!-- the & has to be escaped, since
&copy is a named character reference -->
```

Errors involving known interoperability problems in legacy user agents

Certain syntax constructs are known to cause especially subtle or serious problems in legacy user agents, and are therefore marked as non-conforming to help authors avoid them.

Example For example, this is why the U+0060 GRAVE ACCENT character (`) is not allowed in unquoted attributes. In certain legacy user agents, it is sometimes treated as a quote character.

Example Another example of this is the DOCTYPE, which is required to trigger no-quirks mode, because the behaviour of legacy user agents in quirks mode is often largely undocumented.

Errors that risk exposing authors to security attacks

Certain restrictions exist purely to avoid known security problems.

Example For example, the restriction on using UTF-7 exists purely to avoid authors falling prey to a known cross-site-scripting attack using UTF-7. [UTF7]

Cases where the author's intent is unclear

Markup where the author's intent is very unclear is often made non-conforming. Correcting these errors early makes later maintenance easier.

Example For example, it is unclear whether the author intended the following to be an `h1` heading or an `h2` heading:

```
<h1>Contact details</h2>
```

Cases that are likely to be typos

When a user makes a simple typo, it is helpful if the error can be caught early, as this can save the author a lot of debugging time. This specification therefore usually considers it an error to use element names, attribute names, and so forth, that do not match the names defined in this specification.

Example For example, if the author typed `<capton>` instead of `<caption>`, this would be flagged as an error and the author could correct the typo immediately.

Errors that could interfere with new syntax in the future

In order to allow the language syntax to be extended in the future, certain otherwise harmless features are disallowed.

Example For example, "attributes" in end tags are ignored currently, but they are invalid, in case a future change to the language makes use of that syntax feature without conflicting with already-deployed (and valid!) content.

Some authors find it helpful to be in the practice of always quoting all attributes and always including all optional tags, preferring the consistency derived from such custom over the minor benefits of terseness afforded by making use of the flexibility of the HTML syntax. To aid such authors, conformance checkers can provide modes of operation wherein such conventions are enforced.

File an issue about the selected text

1.12.3 Restrictions on content models and on attribute values

This section is non-normative.

Beyond the syntax of the language, this specification also places restrictions on how elements and attributes can be specified. These restrictions are present for similar reasons:

Errors involving content with dubious semantics

To avoid misuse of elements with defined meanings, content models are defined that restrict how elements can be nested when such nestings would be of dubious value.

Example For example, this specification disallows nesting a `section` element inside a `kbd` element, since it is highly unlikely for an author to indicate that an entire section should be keyed in.

Errors that involve a conflict in expressed semantics

Similarly, to draw the author's attention to mistakes in the use of elements, clear contradictions in the semantics expressed are also considered conformance errors.

Example In the fragments below, for example, the semantics are nonsensical: a separator cannot simultaneously be a cell, nor can a radio button be a progress bar.

```
<hr role="cell">
```

```
<input type=radio role=progressbar>
```

Example Another example is the restrictions on the content models of the `ul` element, which only allows `li` element children. Lists by definition consist just of zero or more list items, so if a `ul` element contains something other than an `li` element, it's not clear what was meant.

Cases where the default styles are likely to lead to confusion

Certain elements have default styles or behaviours that make certain combinations likely to lead to confusion. Where these have equivalent alternatives without this problem, the confusing combinations are disallowed.

Example For example, `div` elements are rendered as block boxes, and `span` elements as inline boxes. Putting a block box in an inline box is unnecessarily confusing; since either nesting just `div` elements, or nesting just `span` elements, or nesting `span` elements inside `div` elements all serve the same purpose as nesting a `div` element in a `span` element, but only the latter involves a block box in an inline box, the latter combination is disallowed.

Example Another example would be the way interactive content cannot be nested. For example, a `button` element cannot contain a `textarea` element. This is because the default behaviour of such nesting interactive elements would be highly confusing to users. Instead of nesting these elements, they can be placed side by side.

Errors that indicate a likely misunderstanding of the specification

Sometimes, something is disallowed because allowing it would likely cause author confusion.

Example For example, setting the `disabled` attribute to the value "`false`" is disallowed, because despite the appearance of meaning that the element is enabled, it in fact means that the element is *disabled* (what matters for implementations is the presence of the attribute, not its value).

Errors involving limits that have been imposed merely to simplify the language

Some conformance errors simplify the language that authors need to learn.

Example For example, the `area` element's `shape` attribute, despite accepting both `circ` and `circle` values in practice as synonyms, disallows the use of the `circ` value, so as to simplify tutorials and other learning aids. There would be no benefit to allowing both, but it would cause extra confusion when teaching the language.

Errors that involve peculiarities of the parser

Certain elements are parsed in somewhat eccentric ways (typically for historical reasons), and their content model restrictions are intended to avoid exposing the author to these issues.

File an issue about the selected text

Example For example, a `form` element isn't allowed inside phrasing content, because when parsed as HTML, a `form` element's start tag will imply a `p` element's end tag. Thus, the following markup results in two paragraphs, not one:

```
<p>Welcome. <form><label>Name:</label> <input></form>
```

It is parsed exactly like the following:

```
<p>Welcome. </p><form><label>Name:</label> <input></form>
```

Errors that would likely result in scripts failing in hard-to-debug ways

Some errors are intended to help prevent script problems that would be hard to debug.

Example This is why, for instance, it is non-conforming to have two `id` attributes with the same value. Duplicate IDs lead to the wrong element being selected, with sometimes disastrous effects whose cause is hard to determine.

Errors that waste authoring time

Some constructs are disallowed because historically they have been the cause of a lot of wasted authoring time, and by encouraging authors to avoid making them, authors can save time in future efforts.

Example For example, a `script` element's `src` attribute causes the element's contents to be ignored. However, this isn't obvious, especially if the element's contents appear to be executable script — which can lead to authors spending a lot of time trying to debug the inline script without realizing that it is not executing. To reduce this problem, this specification makes it non-conforming to have executable script in a `script` element when the `src` attribute is present. This means that authors who are validating their documents are less likely to waste time with this kind of mistake.

Errors that involve areas that affect authors migrating to and from XHTML

Some authors like to write files that can be interpreted as both XML and HTML with similar results. Though this practice is discouraged in general due to the myriad of subtle complications involved (especially when involving scripting, styling, or any kind of automated serialisation), this specification has a few restrictions intended to at least somewhat mitigate the difficulties. This makes it easier for authors to use this as a transitionary step when migrating between HTML and XHTML.

Example For example, there are somewhat complicated rules surrounding the `lang` and `xml:lang` attributes intended to keep the two synchronised.

Example Another example would be the restrictions on the values of `xmlns` attributes in the HTML serialisation, which are intended to ensure that elements in conforming documents end up in the same namespaces whether processed as HTML or XML.

Errors that involve areas reserved for future expansion

As with the restrictions on the syntax intended to allow for new syntax in future revisions of the language, some restrictions on the content models of elements and values of attributes are intended to allow for future expansion of the HTML vocabulary.

Example For example, limiting the values of the `target` attribute that start with an U+005F LOW LINE character (_) to only specific predefined values allows new predefined values to be introduced at a future time without conflicting with author-defined values.

Errors that indicate a mis-use of other specifications

Certain restrictions are intended to support the restrictions made by other specifications.

Example For example, requiring that attributes that take media query lists use only *valid* media query lists reinforces the importance of following the conformance rules of that specification.

File an issue about the selected text

1.13 Suggested reading

This section is non-normative.

The following documents might be of interest to readers of this specification.

Character Model for the World Wide Web 1.0: Fundamentals [CHARMOD]

This Architectural Specification provides authors of specifications, software developers, and content developers with a common reference for interoperable text manipulation on the World Wide Web, building on the Universal Character Set, defined jointly by the Unicode Standard and ISO/IEC 10646. Topics addressed include use of the terms 'character', 'encoding' and 'string', a reference processing model, choice and identification of character encodings, character escaping, and string indexing.

Unicode Security Considerations [UTR36]

Because Unicode contains such a large number of characters and incorporates the varied writing systems of the world, incorrect usage can expose programs or systems to possible security attacks. This is especially important as more and more products are internationalized. This document describes some of the security considerations that programmers, system analysts, standards developers, and users should take into account, and provides specific recommendations to reduce the risk of problems.

Web Content Accessibility Guidelines (WCAG) 2.0 [WCAG]

Web Content Accessibility Guidelines (WCAG) 2.0 covers a wide range of recommendations for making Web content more accessible. Following these guidelines will make content accessible to a wider range of people with disabilities, including blindness and low vision, deafness and hearing loss, learning disabilities, cognitive limitations, limited movement, speech disabilities, photosensitivity and combinations of these. Following these guidelines will also often make your Web content more usable to users in general.

Authoring Tool Accessibility Guidelines (ATAG) 2.0 [ATAG]

This specification provides guidelines for designing Web content authoring tools that are more accessible for people with disabilities. An authoring tool that conforms to these guidelines will promote accessibility by providing an accessible user interface to authors with disabilities as well as by enabling, supporting, and promoting the production of accessible Web content by all authors.

User Agent Accessibility Guidelines (UAAG) 2.0 [UAAG]

This document provides guidelines for designing user agents that lower barriers to Web accessibility for people with disabilities. User agents include browsers and other types of software that retrieve and render Web content. A user agent that conforms to these guidelines will promote accessibility through its own user interface and through other internal facilities, including its ability to communicate with other technologies (especially assistive technologies). Furthermore, all users, not just users with disabilities, should find conforming user agents to be more usable.

Table of Contents — 2 Common infrastructure →

File an issue about the selected text

Extensible Markup Language (XML) 1.0 (Fifth Edition)

W3C Recommendation 26 November 2008

Note: On 7 February 2013, this specification was modified in place to replace broken links to RFC4646 and RFC4647.

This version:
http://www.w3.org/TR/2008/REC-xml-20081126/
Latest version:
http://www.w3.org/TR/xml/
Previous versions:
http://www.w3.org/TR/2008/PER-xml-20080205/
http://www.w3.org/TR/2006/REC-xml-20060816/
Editors:
Tim Bray, Textuality and Netscape <tbray@textuality.com>
Jean Paoli, Microsoft <jeanpa@microsoft.com>
C. M. Sperberg-McQueen, W3C <cmsmcq@w3.org>
Eve Maler, Sun Microsystems, Inc. <eve.maler@east.sun.com>
François Yergeau

Please refer to the **errata** for this document, which may include some normative corrections.

The previous errata for this document, are also available.

See also **translations**.

This document is also available in these non-normative formats: XML and XHTML with color-coded revision indicators.

Abstract

The Extensible Markup Language (XML) is a subset of SGML that is completely described in this document. Its goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML.

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

This document specifies a syntax created by subsetting an existing, widely used international text processing standard (Standard Generalized Markup Language, ISO 8879:1986(E) as amended and corrected) for use on the World Wide Web. It is a product of the XML Core Working Group as part of the XML Activity. The English version of this specification is the only normative version. However, for translations of this document, see http://www.w3.org/2003/03/Translations/byTechnology?technology=xml.

This document is a W3C Recommendation. This fifth edition is *not* a new version of XML. As a convenience to readers, it incorporates the changes dictated by the accumulated errata (available at http://www.w3.org/XML/xml-V10-4e-errata) to the Fourth Edition of XML 1.0, dated 16 August 2006. In particular, erratum [E09] relaxes the restrictions on element and attribute names, thereby providing in XML 1.0 the major end user benefit currently achievable only by using XML 1.1. As a consequence, many possible documents which were not well-formed according to previous editions of this specification are now well-formed, and previously invalid documents using the newly-allowed name characters in, for example, ID attributes, are now valid.

This edition supersedes the previous W3C Recommendation of 16 August 2006.

Please report errors in this document to the public xml-editor@w3.org mail list; public archives are available. For the convenience of readers, an XHTML version with color-coded revision indicators is also provided; this version highlights each change due to an erratum published in the errata list for the previous edition, together with a link to the particular erratum in that list. Most of the errata in the list provide a rationale for the change. The errata list for this fifth edition is available at http://www.w3.org/XML/xml-V10-5e-errata.

An implementation report is available at http://www.w3.org/XML/2008/01/xml10-5e-implementation.html. A Test Suite is maintained to help assessing conformance to this specification.

This document has been reviewed by W3C Members, by software developers, and by other W3C groups and interested parties, and is endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Table of Contents

---

1 Introduction

Extensible Markup Language, abbreviated XML, describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is an application profile or restricted form of SGML, the Standard Generalized Markup Language [ISO 8879]. By construction, XML documents are conforming SGML documents.

XML documents are made up of storage units called entities, which contain either parsed or unparsed data. Parsed data is made up of characters, some of which form character data, and some of which form markup. Markup encodes a description of the document's storage layout and logical structure. XML provides a mechanism to impose constraints on the storage layout and logical structure.

[Definition: A software module called an **XML processor** is used to read XML documents and provide access to their content and structure.] [Definition: It is assumed that an XML processor is doing its work on behalf of another module, called the **application**.] This specification describes the required behavior of an XML processor in terms of how it must read XML data and the information it must provide to the application.

1.1 Origin and Goals

XML was developed by an XML Working Group (originally known as the SGML Editorial Review Board) formed under the auspices of the World Wide Web Consortium (W3C) in 1996. It was chaired by Jon Bosak of Sun Microsystems with the active participation of an XML Special Interest Group (previously known as the SGML Working Group) also organized by the W3C. The membership of the XML Working Group is given in an appendix. Dan Connolly served as the Working Group's contact with the W3C.

The design goals for XML are:

1. XML shall be straightforwardly usable over the Internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.
4. It shall be easy to write programs which process XML documents.
5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly.
8. The design of XML shall be formal and concise.
9. XML documents shall be easy to create.
10. Terseness in XML markup is of minimal importance.

This specification, together with associated standards (Unicode [Unicode] and ISO/IEC 10646 [ISO/IEC 10646] for characters, Internet BCP 47 [IETF BCP 47] and the Language Subtag Registry [IANA-LANGCODES] for language identification tags), provides all the information necessary to understand XML Version 1.0 and construct computer programs to process it.

This version of the XML specification may be distributed freely, as long as all text and legal notices remain intact.

1.2 Terminology

The terminology used to describe XML documents is defined in the body of this specification. The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when EMPHASIZED, are to be interpreted as described in [IETF RFC 2119]. In addition, the terms defined in the following list are used in building those definitions and in describing the actions of an XML processor:

error

[Definition: A violation of the rules of this specification; results are undefined. Unless otherwise specified, failure to observe a prescription of this specification indicated by one of the keywords MUST, REQUIRED, MUST NOT, SHALL and SHALL NOT is an error. Conforming software MAY detect and report an error and MAY recover from it.]

fatal error

[Definition: An error which a conforming XML processor MUST detect and report to the application. After encountering a fatal error, the processor MAY continue processing the data to search for further errors and MAY report such errors to the application. In order to support correction of errors, the processor MAY make unprocessed data from the document (with intermingled character data and markup) available to the application. Once a fatal error is detected, however, the processor MUST NOT continue normal processing (i.e., it MUST NOT continue to pass character data and information about the document's logical structure to the application in the normal way).]

at user option

[Definition: Conforming software MAY or MUST (depending on the modal verb in the sentence) behave as described; if it does, it MUST provide users a means to enable or disable the behavior described.]

validity constraint

[Definition: A rule which applies to all valid XML documents. Violations of validity constraints are errors; they MUST, at user option, be reported by validating XML processors.]

well-formedness constraint

[Definition: A rule which applies to all well-formed XML documents. Violations of well-formedness constraints are fatal errors.]

match

[Definition: (Of strings or names:) Two strings or names being compared are identical. Characters with multiple possible representations in ISO/IEC 10646 (e.g. characters with both precomposed and base+diacritic forms) match only if they have the same representation in both strings. No case folding is performed. (Of strings and rules in the grammar:) A string matches a grammatical production if it belongs to the language generated by that production. (Of content and content models:) An element matches its declaration when it conforms in the fashion described in the constraint **[VC: Element Valid]**.]

for compatibility

[Definition: Marks a sentence describing a feature of XML included solely to ensure that XML remains compatible with SGML.]

for interoperability

[Definition: Marks a sentence describing a non-binding recommendation included to increase the chances that XML documents can be processed by the existing installed base of SGML processors which predate the WebSGML Adaptations Annex to ISO 8879.]

2 Documents

[Definition: A data object is an **XML document** if it is well-formed, as defined in this specification. In addition, the XML document is valid if it meets certain further constraints.]

Each XML document has both a logical and a physical structure. Physically, the document is composed of units called entities. An entity may refer to other entities to cause their inclusion in the document. A document begins in a "root" or document entity. Logically, the document is composed of declarations, elements, comments, character references, and processing instructions, all of which are indicated in the document by explicit markup. The logical and physical structures MUST nest properly, as described in **4.3.2 Well-Formed Parsed Entities**.

2.1 Well-Formed XML Documents

[Definition: A textual object is a **well-formed** XML document if:]

1. Taken as a whole, it matches the production labeled document.
2. It meets all the well-formedness constraints given in this specification.
3. Each of the parsed entities which is referenced directly or indirectly within the document is well-formed.

Document

```
[1]  document  ::=  prolog element Misc*
```

Matching the document production implies that:

1. It contains one or more elements.
2. [Definition: There is exactly one element, called the **root**, or document element, no part of which appears in the content of any other element.] For all other elements, if the start-tag is in the content of another element, the end-tag is in the content of the same element. More simply stated, the elements, delimited by start- and end-tags, nest properly within each other.

[Definition: As a consequence of this, for each non-root element `C` in the document, there is one other element `P` in the document such that `C` is in the content of `P`, but is not in the content of any other element that is in the content of `P`. `P` is referred to as the **parent** of `C`, and `C` as a **child** of `P`.]

2.2 Characters

[Definition: A parsed entity contains **text**, a sequence of characters, which may represent markup or character data.] [Definition: A **character** is an atomic unit of text as specified by ISO/IEC 10646:2000 [ISO/IEC 10646]. Legal characters are tab, carriage return, line feed, and the legal characters of Unicode and ISO/IEC 10646. The versions of these standards cited in **A.1 Normative References** were current at the time this document was prepared. New characters may be added to these standards by amendments or new editions. Consequently, XML processors MUST accept any character in the range specified for Char.]

Character Range

```
[2]  Char  ::=  #x9 | #xA | #xD | [#x20-#xD7FF] |        /* any Unicode character, excluding the surrogate
                [#xE000-#xFFFD] | [#x10000-#x10FFFF]      blocks, FFFE, and FFFF. */
```

The mechanism for encoding character code points into bit patterns may vary from entity to entity. All XML processors MUST accept the UTF-8 and UTF-16 encodings of Unicode [Unicode]; the mechanisms for signaling which of the two is in use, or for bringing other encodings into play, are discussed later, in **4.3.3 Character Encoding in Entities**.

Note:

Document authors are encouraged to avoid "compatibility characters", as defined in section 2.3 of [Unicode]. The characters defined in the following ranges are also discouraged. They are either control characters or permanently undefined Unicode characters:

```
[#x7F-#x84], [#x86-#x9F], [#xFDD0-#xFDEF],
[#x1FFFE-#x1FFFF], [#x2FFFE-#x2FFFF], [#x3FFFE-#x3FFFF],
[#x4FFFE-#x4FFFF], [#x5FFFE-#x5FFFF], [#x6FFFE-#x6FFFF],
[#x7FFFE-#x7FFFF], [#x8FFFE-#x8FFFF], [#x9FFFE-#x9FFFF],
[#xAFFFE-#xAFFFF], [#xBFFFE-#xBFFFF], [#xCFFFE-#xCFFFF],
[#xDFFFE-#xDFFFF], [#xEFFFE-#xEFFFF], [#xFFFFE-#xFFFFF],
[#x10FFFE-#x10FFFF].
```

2.3 Common Syntactic Constructs

This section defines some symbols used widely in the grammar.

S (white space) consists of one or more space (#x20) characters, carriage returns, line feeds, or tabs.

White Space

```
[3]  S  ::=  (#x20 | #x9 | #xD | #xA)+
```

Note:

The presence of #xD in the above production is maintained purely for backward compatibility with the First Edition. As explained in **2.11 End-of-Line Handling**, all #xD characters literally present in an XML document are either removed or replaced by #xA characters before any other processing is done. The only way to get a #xD character to match this production is to use a character reference in an entity value literal.

An Nmtoken (name token) is any mixture of name characters.

[Definition: A Name is an Nmtoken with a restricted set of initial characters.] Disallowed initial characters for Names include digits, diacritics, the full stop and the hyphen.

Names beginning with the string "`xml`", or with any string which would match `(('X'|'x') ('M'|'m') ('L'|'l'))`, are reserved for standardization in this or future versions of this specification.

Note:

The Namespaces in XML Recommendation [XML Names] assigns a meaning to names containing colon characters. Therefore, authors should not use the colon in XML names except for namespace purposes, but XML processors must accept the colon as a name character.

The first character of a Name MUST be a NameStartChar, and any other characters MUST be NameChars; this mechanism is used to prevent names from beginning with European (ASCII) digits or with basic combining characters. Almost all characters are permitted in names, except those which either are or reasonably could be used as delimiters. The intention is to be inclusive rather than exclusive, so that writing systems not yet encoded in Unicode can be used in XML names. See **J Suggestions for XML Names** for suggestions on the creation of names.

Document authors are encouraged to use names which are meaningful words or combinations of words in natural languages, and to avoid symbolic or white space characters in names. Note that COLON, HYPHEN-MINUS, FULL STOP (period), LOW LINE (underscore), and MIDDLE DOT are explicitly permitted.

The ASCII symbols and punctuation marks, along with a fairly large group of Unicode symbol characters, are excluded from names because they are more useful as delimiters in contexts where XML names are used outside XML documents; providing this group gives those contexts hard guarantees about what *cannot* be part of an XML name. The character #x037E, GREEK QUESTION MARK, is excluded because when normalized it becomes a semicolon, which could change the meaning of entity references.

Names and Tokens

```
[4]   NameStartChar  ::=  ":" | [A-Z] | "_" | [a-z] | [#xC0-#xD6] | [#xD8-#xF6] | [#xF8-#x2FF] |
                          [#x370-#x37D] | [#x37F-#x1FFF] | [#x200C-#x200D] | [#x2070-#x218F] |
                          [#x2C00-#x2FEF] | [#x3001-#xD7FF] | [#xF900-#xFDCF] | [#xFDF0-#xFFFD] |
                          [#x10000-#xEFFFF]
[4a]  NameChar       ::=  NameStartChar | "-" | "." | [0-9] | #xB7 | [#x0300-#x036F] | [#x203F-#x2040]
[5]   Name           ::=  NameStartChar (NameChar)*
[6]   Names          ::=  Name (#x20 Name)*
[7]   Nmtoken        ::=  (NameChar)+
[8]   Nmtokens       ::=  Nmtoken (#x20 Nmtoken)*
```

Note:

The Names and Nmtokens productions are used to define the validity of tokenized attribute values after normalization (see **3.3.1 Attribute Types**).

Literal data is any quoted string not containing the quotation mark used as a delimiter for that string. Literals are used for specifying the content of internal entities (EntityValue), the values of attributes (AttValue), and external identifiers (SystemLiteral). Note that a SystemLiteral can be parsed without scanning for markup.

Literals

```
[9]   EntityValue    ::=  '"' ([^%&"] | PEReference | Reference)* '"'
                          |  "'" ([^%&'] | PEReference | Reference)* "'"
[10]  AttValue       ::=  '"' ([^<&"] | Reference)* '"'
                          |  "'" ([^<&'] | Reference)* "'"
[11]  SystemLiteral  ::=  ('"' [^"]* '"') | ("'" [^']* "'")
[12]  PubidLiteral   ::=  '"' PubidChar* '"' | "'" (PubidChar - "'")* "'"
[13]  PubidChar      ::=  #x20 | #xD | #xA | [a-zA-Z0-9] | [-'()+,./:=?;!*#@$_%]
```

Note:

Although the EntityValue production allows the definition of a general entity consisting of a single explicit `<` in the literal (e.g., `<!ENTITY mylt "<">`), it is strongly advised to avoid this practice since any reference to that entity will cause a well-formedness error.

2.4 Character Data and Markup

Text consists of intermingled character data and markup. [Definition: **Markup** takes the form of start-tags, end-tags, empty-element tags, entity references, character references, comments, CDATA section delimiters, document type declarations, processing instructions, XML declarations, text declarations, and any white space that is at the top level of the document entity (that is, outside the document element and not inside any other markup).]

[Definition: All text that is not markup constitutes the **character data** of the document.]

The ampersand character (&) and the left angle bracket (<) MUST NOT appear in their literal form, except when used as markup delimiters, or within a comment, a processing instruction, or a CDATA section. If they are needed elsewhere, they MUST be escaped using either numeric character references or the strings " `&amp;` " and " `&lt;` " respectively. The right angle bracket (>) may be represented using the string " `&gt;` ", and MUST, for compatibility, be escaped using either " `&gt;` " or a character reference when it appears in the string " `]]>` " in content, when that string is not marking the end of a CDATA section.

In the content of elements, character data is any string of characters which does not contain the start-delimiter of any markup and does not include the CDATA-section-close delimiter, " `]]>` ". In a CDATA section, character data is any string of characters not including the CDATA-section-close delimiter, " `]]>` ".

To allow attribute values to contain both single and double quotes, the apostrophe or single-quote character (') may be represented as " `&apos;` ", and the double-quote character (") as " `&quot;` ".

Character Data

```
[14]  CharData  ::=  [^<&]* - ([^<&]* ']]>' [^<&]*)
```

2.5 Comments

[Definition: **Comments** may appear anywhere in a document outside other markup; in addition, they may appear within the document type declaration at places allowed by the grammar. They are not part of the document's character data; an XML processor MAY, but need not, make it possible for an application to retrieve the text of comments. For compatibility, the string " `--` " (double-hyphen) MUST NOT occur within comments.] Parameter entity references MUST NOT be recognized within comments.

Comments

```
[15]  Comment  ::=  '<!--' ((Char - '-') | ('-' (Char - '-')))* '-->'
```

An example of a comment:

```
<!-- declarations for <head> & <body> -->
```

Note that the grammar does not allow a comment ending in `--->`. The following example is *not* well-formed.

```
<!-- B+, B, or B--->
```

2.6 Processing Instructions

[Definition: **Processing instructions** (PIs) allow documents to contain instructions for applications.]

Processing Instructions

```
[16]  PI        ::=  '<?' PITarget (S (Char* - (Char* '?>' Char*)))? '?>'
[17]  PITarget  ::=  Name - (('X' | 'x') ('M' | 'm') ('L' | 'l'))
```

PIs are not part of the document's character data, but MUST be passed through to the application. The PI begins with a target (PITarget) used to identify the application to which the instruction is directed. The target names " `XML` ", " `xml` ", and so on are reserved for standardization in this or future versions of this specification. The XML Notation mechanism may be used for formal declaration of PI targets. Parameter entity references MUST NOT be recognized within processing instructions.

2.7 CDATA Sections

[Definition: **CDATA sections** may occur anywhere character data may occur; they are used to escape blocks of text containing characters which would otherwise be recognized as markup. CDATA sections begin with the string " `<![CDATA[` " and end with the string " `]]>` ":]

CDATA Sections

[18]	`CDSect`	::=	`CDStart CData CDEnd`
[19]	`CDStart`	::=	`'<![CDATA['`
[20]	`CData`	::=	`(Char* - (Char* ']]>' Char*))`
[21]	`CDEnd`	::=	`']]>'`

Within a CDATA section, only the CDEnd string is recognized as markup, so that left angle brackets and ampersands may occur in their literal form; they need not (and cannot) be escaped using " `&lt;` " and " `&amp;` ". CDATA sections cannot nest.

An example of a CDATA section, in which " `<greeting>` " and " `</greeting>` " are recognized as character data, not markup:

```
<![CDATA[<greeting>Hello, world!</greeting>]]>
```

2.8 Prolog and Document Type Declaration

[Definition: XML documents SHOULD begin with an **XML declaration** which specifies the version of XML being used.] For example, the following is a complete XML document, well-formed but not valid:

```
<?xml version="1.0"?>
<greeting>Hello, world!</greeting>
```

and so is this:

```
<greeting>Hello, world!</greeting>
```

The function of the markup in an XML document is to describe its storage and logical structure and to associate attribute name-value pairs with its logical structures. XML provides a mechanism, the document type declaration, to define constraints on the logical structure and to support the use of predefined storage units. [Definition: An XML document is **valid** if it has an associated document type declaration and if the document complies with the constraints expressed in it.]

The document type declaration MUST appear before the first element in the document.

Prolog

[22]	`prolog`	::=	`XMLDecl? Misc* (doctypedecl Misc*)?`
[23]	`XMLDecl`	::=	`'<?xml' VersionInfo EncodingDecl? SDDecl? S? '?>'`
[24]	`VersionInfo`	::=	`S 'version' Eq ("'" VersionNum "'" | '"' VersionNum '"')`
[25]	`Eq`	::=	`S? '=' S?`
[26]	`VersionNum`	::=	`'1.' [0-9]+`
[27]	`Misc`	::=	`Comment | PI | S`

Even though the VersionNum production matches any version number of the form '1.x', XML 1.0 documents SHOULD NOT specify a version number other than '1.0'.

Note:

When an XML 1.0 processor encounters a document that specifies a 1.x version number other than '1.0', it will process it as a 1.0 document. This means that an XML 1.0 processor will accept 1.x documents provided they do not use any non-1.0 features.

[Definition: The XML **document type declaration** contains or points to markup declarations that provide a grammar for a class of documents. This grammar is known as a document type definition, or **DTD**. The document type declaration can point to an external subset (a special kind of external entity) containing markup declarations, or can contain the markup declarations directly in an internal subset, or can do both. The DTD for a document consists of both subsets taken together.]

[Definition: A **markup declaration** is an element type declaration, an attribute-list declaration, an entity declaration, or a notation declaration.] These declarations may be contained in whole or in part within parameter entities, as described in the well-formedness and validity constraints below. For further information, see **4 Physical Structures**.

Document Type Definition

[28]	`doctypedecl`	::=	`'<!DOCTYPE' S Name (S ExternalID)? S? ('[' intSubset ']' S?)? '>'`	[VC: Root Element Type] [WFC: External Subset]
[28a]	`DeclSep`	::=	`PEReference | S`	[WFC: PE Between Declarations]
[28b]	`intSubset`	::=	`(markupdecl | DeclSep)*`	
[29]	`markupdecl`	::=	`elementdecl | AttlistDecl | EntityDecl | NotationDecl | PI | Comment`	[VC: Proper Declaration/PE Nesting] [WFC: PEs in Internal Subset]

Note that it is possible to construct a well-formed document containing a doctypedecl that neither points to an external subset nor contains an internal subset.

The markup declarations may be made up in whole or in part of the replacement text of parameter entities. The productions later in this specification for individual nonterminals (elementdecl, AttlistDecl, and so on) describe the declarations *after* all the parameter entities have been included.

Parameter entity references are recognized anywhere in the DTD (internal and external subsets and external parameter entities), except in literals, processing instructions, comments, and the contents of ignored conditional sections (see **3.4 Conditional Sections**). They are also recognized in entity value literals. The use of parameter entities in the internal subset is restricted as described below.

Validity constraint: Root Element Type

The Name in the document type declaration MUST match the element type of the root element.

Validity constraint: Proper Declaration/PE Nesting

Parameter-entity replacement text MUST be properly nested with markup declarations. That is to say, if either the first character or the last character of a markup declaration (markupdecl above) is contained in the replacement text for a parameter-entity reference, both MUST be contained in the same replacement text.

Well-formedness constraint: PEs in Internal Subset

In the internal DTD subset, parameter-entity references MUST NOT occur within markup declarations; they may occur where markup declarations can occur. (This does not apply to references that occur in external parameter entities or to the external subset.)

Well-formedness constraint: External Subset

The external subset, if any, MUST match the production for extSubset.

Well-formedness constraint: PE Between Declarations

The replacement text of a parameter entity reference in a DeclSep MUST match the production extSubsetDecl.

Like the internal subset, the external subset and any external parameter entities referenced in a DeclSep MUST consist of a series of complete markup declarations of the types allowed by the non-terminal symbol markupdecl, interspersed with white space or parameter-entity references. However, portions of the contents of the external subset or of these external parameter entities may conditionally be ignored by using the conditional section construct; this is not allowed in the internal subset but is allowed in external parameter entities referenced in the internal subset.

External Subset

[30]	`extSubset`	::=	`TextDecl? extSubsetDecl`
[31]	`extSubsetDecl`	::=	`( markupdecl | conditionalSect | DeclSep)*`

The external subset and external parameter entities also differ from the internal subset in that in them, parameter-entity references are permitted *within* markup declarations, not only *between* markup declarations.

An example of an XML document with a document type declaration:

```
<?xml version="1.0"?>
<!DOCTYPE greeting SYSTEM "hello.dtd">
<greeting>Hello, world!</greeting>
```

The system identifier " `hello.dtd` " gives the address (a URI reference) of a DTD for the document.

The declarations can also be given locally, as in this example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE greeting [
  <!ELEMENT greeting (#PCDATA)>
]>
<greeting>Hello, world!</greeting>
```

If both the external and internal subsets are used, the internal subset MUST be considered to occur before the external subset. This has the effect that entity and attribute-list declarations in the internal subset take precedence over those in the external subset.

2.9 Standalone Document Declaration

Markup declarations can affect the content of the document, as passed from an XML processor to an application; examples are attribute defaults and entity declarations. The standalone document declaration, which may appear as a component of the XML declaration, signals whether or not there are such declarations which appear external to the document entity or in parameter entities. [Definition: An **external markup declaration** is defined as a markup declaration occurring in the external subset or in a parameter entity (external or internal, the latter being included because non-validating processors are not required to read them).]

Standalone Document Declaration

[32]	`SDDecl`	::=	`S 'standalone' Eq (("'" ('yes' | 'no') "'") | ('"' ('yes' | 'no') '"'))`	[VC: Standalone Document Declaration]

In a standalone document declaration, the value "yes" indicates that there are no external markup declarations which affect the information passed from the XML processor to the application. The value "no" indicates that there are or may be such external markup declarations. Note that the standalone document declaration only denotes the presence of external *declarations*; the presence, in a document, of references to external *entities*, when those entities are internally declared, does not change its standalone status.

If there are no external markup declarations, the standalone document declaration has no meaning. If there are external markup declarations but there is no standalone document declaration, the value "no" is assumed.

Any XML document for which `standalone="no"` holds can be converted algorithmically to a standalone document, which may be desirable for some network delivery applications.

Validity constraint: Standalone Document Declaration

The standalone document declaration MUST have the value "no" if any external markup declarations contain declarations of:

- attributes with default values, if elements to which these attributes apply appear in the document without specifications of values for these attributes, or
- entities (other than `amp`, `lt`, `gt`, `apos`, `quot`), if references to those entities appear in the document, or
- attributes with tokenized types, where the attribute appears in the document with a value such that *normalization* will produce a different value from that which would be produced in the absence of the declaration, or
- element types with element content, if white space occurs directly within any instance of those types.

An example XML declaration with a standalone document declaration:

```
<?xml version="1.0" standalone='yes'?>
```

2.10 White Space Handling

In editing XML documents, it is often convenient to use "white space" (spaces, tabs, and blank lines) to set apart the markup for greater readability. Such white space is typically not intended for inclusion in the delivered version of the document. On the other hand, "significant" white space that should be preserved in the delivered version is common, for example in poetry and source code.

An XML processor MUST always pass all characters in a document that are not markup through to the application. A validating XML processor MUST also inform the application which of these characters constitute white space appearing in element content.

A special attribute named `xml:space` may be attached to an element to signal an intention that in that element, white space should be preserved by applications. In valid documents, this attribute, like any other, MUST be declared if it is used. When declared, it MUST be given as an enumerated type whose values are one or both of "default" and "preserve". For example:

```
<!ATTLIST poem  xml:space (default|preserve) 'preserve'>

<!ATTLIST pre xml:space (preserve) #FIXED 'preserve'>
```

The value "default" signals that applications' default white-space processing modes are acceptable for this element; the value "preserve" indicates the intent that applications preserve all the white space. This declared intent is considered to apply to all elements within the content of the element where it is specified, unless overridden with another instance of the `xml:space` attribute. This specification does not give meaning to any value of `xml:space` other than "default" and "preserve". It is an error for other values to be specified; the XML processor MAY report the error or MAY recover by ignoring the attribute specification or by reporting the (erroneous) value to the application. Applications may ignore or reject erroneous values.

The root element of any document is considered to have signaled no intentions as regards application space handling, unless it provides a value for this attribute or the attribute is declared with a default value.

2.11 End-of-Line Handling

XML parsed entities are often stored in computer files which, for editing convenience, are organized into lines. These lines are typically separated by some combination of the characters CARRIAGE RETURN (#xD) and LINE FEED (#xA).

To simplify the tasks of applications, the XML processor MUST behave as if it normalized all line breaks in external parsed entities (including the document entity) on input, before parsing, by translating both the two-character sequence #xD #xA and any #xD that is not followed by #xA to a single #xA character.

2.12 Language Identification

In document processing, it is often useful to identify the natural or formal language in which the content is written. A special attribute named `xml:lang` may be inserted in documents to specify the language used in the contents and attribute values of any element in an XML document. In valid documents, this attribute, like any other, MUST be declared if it is used. The values of the attribute are language identifiers as defined by [IETF BCP 47], *Tags for the Identification of Languages*; in addition, the empty string may be specified.

(Productions 33 through 38 have been removed.)

For example:

```
<p xml:lang="en">The quick brown fox jumps over the lazy dog.</p>
<p xml:lang="en-GB">What colour is it?</p>
<p xml:lang="en-US">What color is it?</p>
<sp who="Faust" desc='leise' xml:lang="de">
  <l>Habe nun, ach! Philosophie,</l>
  <l>Juristerei, und Medizin</l>
  <l>und leider auch Theologie</l>
  <l>durchaus studiert mit heißem Bemüh'n.</l>
</sp>
```

The language specified by `xml:lang` applies to the element where it is specified (including the values of its attributes), and to all elements in its content unless overridden with another instance of `xml:lang`. In particular, the empty value of `xml:lang` is used on an element B to override a specification of `xml:lang` on an enclosing element A, without specifying another language. Within B, it is considered that there is no language information available, just as if `xml:lang` had not been specified on B or any of its ancestors. Applications determine which of an element's attribute values and which parts of its character content, if any, are treated as language-dependent values described by `xml:lang`.

Note:

Language information may also be provided by external transport protocols (e.g. HTTP or MIME). When available, this information may be used by XML applications, but the more local information provided by `xml:lang` should be considered to override it.

A simple declaration for `xml:lang` might take the form

```
  xml:lang CDATA #IMPLIED
```

but specific default values may also be given, if appropriate. In a collection of French poems for English students, with glosses and notes in English, the `xml:lang` attribute might be declared this way:

```
  <!ATTLIST poem   xml:lang CDATA 'fr'>
  <!ATTLIST gloss  xml:lang CDATA 'en'>
  <!ATTLIST note   xml:lang CDATA 'en'>
```

3 Logical Structures

[Definition: Each XML document contains one or more **elements**, the boundaries of which are either delimited by start-tags and end-tags, or, for empty elements, by an empty-element tag. Each element has a type, identified by name, sometimes called its "generic identifier" (GI), and may have a set of attribute specifications.] Each attribute specification has a name and a value.

Element

```
[39]  element  ::=  EmptyElemTag
                    | STag content ETag [WFC: Element Type Match]
                                        [VC: Element Valid]
```

This specification does not constrain the application semantics, use, or (beyond syntax) names of the element types and attributes, except that names beginning with a match to `(('X'|'x')('M'|'m')('L'|'l'))` are reserved for standardization in this or future versions of this specification.

Well-formedness constraint: Element Type Match

The Name in an element's end-tag MUST match the element type in the start-tag.

Validity constraint: Element Valid

An element is valid if there is a declaration matching elementdecl where the Name matches the element type, and one of the following holds:

1. The declaration matches **EMPTY** and the element has no content (not even entity references, comments, PIs or white space).
2. The declaration matches children and the sequence of child elements belongs to the language generated by the regular expression in the content model, with optional white space, comments and PIs (i.e. markup matching production [27] Misc) between the start-tag and the first child element, between child elements, or between the last child element and the end-tag. Note that a CDATA section containing only white space or a reference to an entity whose replacement text is character references expanding to white space do not match the nonterminal S, and hence cannot appear in these positions; however, a reference to an internal entity with a literal value consisting of character references expanding to white space does match S, since its replacement text is the white space resulting from expansion of the character references.
3. The declaration matches Mixed, and the content (after replacing any entity references with their replacement text) consists of character data (including CDATA sections), comments, PIs and child elements whose types match names in the content model.
4. The declaration matches **ANY**, and the content (after replacing any entity references with their replacement text) consists of character data, CDATA sections, comments, PIs and child elements whose types have been declared.

3.1 Start-Tags, End-Tags, and Empty-Element Tags

[Definition: The beginning of every non-empty XML element is marked by a **start-tag**.]

Start-tag

```
[40]  STag       ::=  '<' Name (S Attribute)* S? '>' [WFC: Unique Att Spec]
[41]  Attribute  ::=  Name Eq AttValue               [VC: Attribute Value Type]
                                                     [WFC: No External Entity References]
                                                     [WFC: No < in Attribute Values]
```

The Name in the start- and end-tags gives the element's **type**. [Definition: The Name-AttValue pairs are referred to as the **attribute specifications** of the element], [Definition: with the Name in each pair referred to as the **attribute name**] and [Definition: the content of the AttValue (the text between the `'` or `"` delimiters) as the **attribute value**.] Note that the order of attribute specifications in a start-tag or empty-element tag is not significant.

Well-formedness constraint: Unique Att Spec

An attribute name MUST NOT appear more than once in the same start-tag or empty-element tag.

Validity constraint: Attribute Value Type

The attribute MUST have been declared; the value MUST be of the type declared for it. (For attribute types, see **3.3 Attribute-List Declarations**.)

Well-formedness constraint: No External Entity References

Attribute values MUST NOT contain direct or indirect entity references to external entities.

Well-formedness constraint: No `<` in Attribute Values

The replacement text of any entity referred to directly or indirectly in an attribute value MUST NOT contain a `<`.

An example of a start-tag:

```
<termdef id="dt-dog" term="dog">
```

[Definition: The end of every element that begins with a start-tag MUST be marked by an **end-tag** containing a name that echoes the element's type as given in the start-tag:]

End-tag

[42] `ETag ::= '</' Name S? '>'`

An example of an end-tag:

```
</termdef>
```

[Definition: The text between the start-tag and end-tag is called the element's **content**:]

Content of Elements

[43] `content ::= CharData? ((element | Reference | CDSect | PI | Comment) CharData?)*`

[Definition: An element with no content is said to be **empty**.] The representation of an empty element is either a start-tag immediately followed by an end-tag, or an empty-element tag. [Definition: An **empty-element tag** takes a special form:]

Tags for Empty Elements

[44] `EmptyElemTag ::= '<' Name (S Attribute)* S? '/>'` [WFC: Unique Att Spec]

Empty-element tags may be used for any element which has no content, whether or not it is declared using the keyword **EMPTY**. For interoperability, the empty-element tag SHOULD be used, and SHOULD only be used, for elements which are declared EMPTY.

Examples of empty elements:

```
<IMG align="left"
 src="http://www.w3.org/Icons/WWW/w3c_home" />
<br></br>
<br/>
```

3.2 Element Type Declarations

The element structure of an XML document may, for validation purposes, be constrained using element type and attribute-list declarations. An element type declaration constrains the element's content.

Element type declarations often constrain which element types can appear as children of the element. At user option, an XML processor MAY issue a warning when a declaration mentions an element type for which no declaration is provided, but this is not an error.

[Definition: An **element type declaration** takes the form:]

Element Type Declaration

[45]	elementdecl	::=	'<!ELEMENT' S Name S contentspec S? '>'	[VC: Unique Element Type Declaration]
[46]	contentspec	::=	'EMPTY' | 'ANY' | Mixed | children	

where the Name gives the element type being declared.

Validity constraint: Unique Element Type Declaration

An element type MUST NOT be declared more than once.

Examples of element type declarations:

```
<!ELEMENT br EMPTY>
<!ELEMENT p (#PCDATA|emph)* >
<!ELEMENT %name.para; %content.para; >
<!ELEMENT container ANY>
```

3.2.1 Element Content

[Definition: An element type has **element content** when elements of that type MUST contain only child elements (no character data), optionally separated by white space (characters matching the nonterminal S).] [Definition: In this case, the constraint includes a **content model**, a simple grammar governing the allowed types of the child elements and the order in which they are allowed to appear.] The grammar is built on content particles (cps), which consist of names, choice lists of content particles, or sequence lists of content particles:

Element-content Models

[47]	children	::=	(choice | seq) ('?' | '*' | '+')?	
[48]	cp	::=	(Name | choice | seq) ('?' | '*' | '+')?	
[49]	choice	::=	'(' S? cp (S? '|' S? cp)+ S? ')'	[VC: Proper Group/PE Nesting]
[50]	seq	::=	'(' S? cp (S? ',' S? cp)* S? ')'	[VC: Proper Group/PE Nesting]

where each Name is the type of an element which may appear as a child. Any content particle in a choice list may appear in the element content at the location where the choice list appears in the grammar; content particles occurring in a sequence list MUST each appear in the element content in the order given in the list. The optional character following a name or list governs whether the element or the content particles in the list may occur one or more (+), zero or more (*), or zero or one times (?). The absence of such an operator means that the element or content particle MUST appear exactly once. This syntax and meaning are identical to those used in the productions in this specification.

The content of an element matches a content model if and only if it is possible to trace out a path through the content model, obeying the sequence, choice, and repetition operators and matching each element in the content against an element type in the content model. For compatibility, it is an error if the content model allows an element to match more than one occurrence of an element type in the content model. For more information, see **E Deterministic Content Models**.

Validity constraint: Proper Group/PE Nesting

Parameter-entity replacement text MUST be properly nested with parenthesized groups. That is to say, if either of the opening or closing parentheses in a choice, seq, or Mixed construct is contained in the replacement text for a parameter entity, both MUST be contained in the same replacement text.

For interoperability, if a parameter-entity reference appears in a choice, seq, or Mixed construct, its replacement text SHOULD contain at least one non-blank character, and neither the first nor last non-blank character of the replacement text SHOULD be a connector (| or ,).

Examples of element-content models:

```
<!ELEMENT spec (front, body, back?)>
<!ELEMENT div1 (head, (p | list | note)*, div2*)>
<!ELEMENT dictionary-body (%div.mix; | %dict.mix;)*>
```

3.2.2 Mixed Content

[Definition: An element type has **mixed content** when elements of that type may contain character data, optionally interspersed with child elements.] In this case, the types of the child elements may be constrained, but not their order or their number of occurrences:

Mixed-content Declaration

```
[51]  Mixed  ::=  '(' S? '#PCDATA' (S? '|' S? Name)* S? ')*'
                  | '(' S? '#PCDATA' S? ')'              [VC: Proper Group/PE Nesting]
                                                          [VC: No Duplicate Types]
```

where the Names give the types of elements that may appear as children. The keyword **#PCDATA** derives historically from the term "parsed character data."

Validity constraint: No Duplicate Types

The same name MUST NOT appear more than once in a single mixed-content declaration.

Examples of mixed content declarations:

```
<!ELEMENT p (#PCDATA|a|ul|b|i|em)*>
<!ELEMENT p (#PCDATA | %font; | %phrase; | %special; | %form;)* >
<!ELEMENT b (#PCDATA)>
```

3.3 Attribute-List Declarations

Attributes are used to associate name-value pairs with elements. Attribute specifications MUST NOT appear outside of start-tags and empty-element tags; thus, the productions used to recognize them appear in **3.1 Start-Tags, End-Tags, and Empty-Element Tags**. Attribute-list declarations may be used:

- To define the set of attributes pertaining to a given element type.
- To establish type constraints for these attributes.
- To provide default values for attributes.

[Definition: **Attribute-list declarations** specify the name, data type, and default value (if any) of each attribute associated with a given element type:]

Attribute-list Declaration

```
[52]  AttlistDecl  ::=  '<!ATTLIST' S Name AttDef* S? '>'
[53]  AttDef       ::=  S Name S AttType S DefaultDecl
```

The Name in the AttlistDecl rule is the type of an element. At user option, an XML processor MAY issue a warning if attributes are declared for an element type not itself declared, but this is not an error. The Name in the AttDef rule is the name of the attribute.

When more than one AttlistDecl is provided for a given element type, the contents of all those provided are merged. When more than one definition is provided for the same attribute of a given element type, the first declaration is binding and later declarations are ignored. For interoperability, writers of DTDs may choose to provide at most one attribute-list declaration for a given element type, at most one attribute definition for a given attribute name in an attribute-list declaration, and at least one attribute definition in each attribute-list declaration. For interoperability, an XML processor MAY at user option issue a warning when more than one attribute-list declaration is provided for a given element type, or more than one attribute definition is provided for a given attribute, but this is not an error.

3.3.1 Attribute Types

XML attribute types are of three kinds: a string type, a set of tokenized types, and enumerated types. The string type may take any literal string as a value; the tokenized types are more constrained. The validity constraints noted in the grammar are applied after the attribute value has been normalized as described in **3.3.3 Attribute-Value Normalization**.

Attribute Types

```
[54]  AttType  ::=  StringType | TokenizedType | EnumeratedType
```

[55]	StringType	::=	'CDATA'	
[56]	TokenizedType	::=	'ID'	[VC: ID] [VC: One ID per Element Type] [VC: ID Attribute Default]
			| 'IDREF'	[VC: IDREF]
			| 'IDREFS'	[VC: IDREF]
			| 'ENTITY'	[VC: Entity Name]
			| 'ENTITIES'	[VC: Entity Name]
			| 'NMTOKEN'	[VC: Name Token]
			| 'NMTOKENS'	[VC: Name Token]

Validity constraint: ID

Values of type **ID** MUST match the Name production. A name MUST NOT appear more than once in an XML document as a value of this type; i.e., ID values MUST uniquely identify the elements which bear them.

Validity constraint: One ID per Element Type

An element type MUST NOT have more than one ID attribute specified.

Validity constraint: ID Attribute Default

An ID attribute MUST have a declared default of **#IMPLIED** or **#REQUIRED**.

Validity constraint: IDREF

Values of type **IDREF** MUST match the Name production, and values of type **IDREFS** MUST match Names; each Name MUST match the value of an ID attribute on some element in the XML document; i.e. **IDREF** values MUST match the value of some ID attribute.

Validity constraint: Entity Name

Values of type **ENTITY** MUST match the Name production, values of type **ENTITIES** MUST match Names; each Name MUST match the name of an unparsed entity declared in the DTD.

Validity constraint: Name Token

Values of type **NMTOKEN** MUST match the Nmtoken production; values of type **NMTOKENS** MUST match Nmtokens.

[Definition: **Enumerated attributes** have a list of allowed values in their declaration]. They MUST take one of those values. There are two kinds of enumerated attribute types:

Enumerated Attribute Types

[57]	EnumeratedType	::=	NotationType | Enumeration	
[58]	NotationType	::=	'NOTATION' S '(' S? Name (S? '|' S? Name)* S? ')'	[VC: Notation Attributes] [VC: One Notation Per Element Type] [VC: No Notation on Empty Element] [VC: No Duplicate Tokens]
[59]	Enumeration	::=	'(' S? Nmtoken (S? '|' S? Nmtoken)* S? ')'	[VC: Enumeration] [VC: No Duplicate Tokens]

A **NOTATION** attribute identifies a notation, declared in the DTD with associated system and/or public identifiers, to be used in interpreting the element to which the attribute is attached.

Validity constraint: Notation Attributes

Values of this type MUST match one of the *notation* names included in the declaration; all notation names in the declaration MUST be declared.

Validity constraint: One Notation Per Element Type

An element type MUST NOT have more than one **NOTATION** attribute specified.

Validity constraint: No Notation on Empty Element

For compatibility, an attribute of type **NOTATION** MUST NOT be declared on an element declared **EMPTY**.

Validity constraint: No Duplicate Tokens

The notation names in a single NotationType attribute declaration, as well as the NmTokens in a single Enumeration attribute declaration, MUST all be distinct.

Validity constraint: Enumeration

Values of this type MUST match one of the Nmtoken tokens in the declaration.

For interoperability, the same Nmtoken SHOULD NOT occur more than once in the enumerated attribute types of a single element type.

3.3.2 Attribute Defaults

An attribute declaration provides information on whether the attribute's presence is REQUIRED, and if not, how an XML processor is to react if a declared attribute is absent in a document.

Attribute Defaults

```
[60]  DefaultDecl  ::=  '#REQUIRED' | '#IMPLIED'
                        | (('#FIXED' S)? AttValue) [VC: Required Attribute]
                                                   [VC: Attribute Default Value Syntactically Correct]
                                                   [WFC: No < in Attribute Values]
                                                   [VC: Fixed Attribute Default]
                                                   [WFC: No External Entity References]
```

In an attribute declaration, **#REQUIRED** means that the attribute MUST always be provided, **#IMPLIED** that no default value is provided. [Definition: If the declaration is neither **#REQUIRED** nor **#IMPLIED**, then the AttValue value contains the declared **default** value; the **#FIXED** keyword states that the attribute MUST always have the default value. When an XML processor encounters an element without a specification for an attribute for which it has read a default value declaration, it MUST report the attribute with the declared default value to the application.]

Validity constraint: Required Attribute

If the default declaration is the keyword **#REQUIRED**, then the attribute MUST be specified for all elements of the type in the attribute-list declaration.

Validity constraint: Attribute Default Value Syntactically Correct

The declared default value MUST meet the syntactic constraints of the declared attribute type. That is, the default value of an attribute:

- of type IDREF or ENTITY must match the Name production;
- of type IDREFS or ENTITIES must match the Names production;
- of type NMTOKEN must match the Nmtoken production;
- of type NMTOKENS must match the Nmtokens production;
- of an enumerated type (either a NOTATION type or an enumeration) must match one of the enumerated values.

Note that only the syntactic constraints of the type are required here; other constraints (e.g. that the value be the name of a declared unparsed entity, for an attribute of type ENTITY) will be reported by a validating parser only if an element without a specification for this attribute actually occurs.

Validity constraint: Fixed Attribute Default

If an attribute has a default value declared with the **#FIXED** keyword, instances of that attribute MUST match the default value.

Examples of attribute-list declarations:

```
<!ATTLIST termdef
          id      ID      #REQUIRED
          name    CDATA   #IMPLIED>
<!ATTLIST list
          type    (bullets|ordered|glossary)  "ordered">
```

```
<!ATTLIST form
          method  CDATA   #FIXED "POST">
```

3.3.3 Attribute-Value Normalization

Before the value of an attribute is passed to the application or checked for validity, the XML processor MUST normalize the attribute value by applying the algorithm below, or by using some other method such that the value passed to the application is the same as that produced by the algorithm.

1. All line breaks MUST have been normalized on input to #xA as described in **2.11 End-of-Line Handling**, so the rest of this algorithm operates on text normalized in this way.
2. Begin with a normalized value consisting of the empty string.
3. For each character, entity reference, or character reference in the unnormalized attribute value, beginning with the first and continuing to the last, do the following:
    - For a character reference, append the referenced character to the normalized value.
    - For an entity reference, recursively apply step 3 of this algorithm to the replacement text of the entity.
    - For a white space character (#x20, #xD, #xA, #x9), append a space character (#x20) to the normalized value.
    - For another character, append the character to the normalized value.

If the attribute type is not CDATA, then the XML processor MUST further process the normalized attribute value by discarding any leading and trailing space (#x20) characters, and by replacing sequences of space (#x20) characters by a single space (#x20) character.

Note that if the unnormalized attribute value contains a character reference to a white space character other than space (#x20), the normalized value contains the referenced character itself (#xD, #xA or #x9). This contrasts with the case where the unnormalized value contains a white space character (not a reference), which is replaced with a space character (#x20) in the normalized value and also contrasts with the case where the unnormalized value contains an entity reference whose replacement text contains a white space character; being recursively processed, the white space character is replaced with a space character (#x20) in the normalized value.

All attributes for which no declaration has been read SHOULD be treated by a non-validating processor as if declared **CDATA**.

It is an error if an attribute value contains a reference to an entity for which no declaration has been read.

Following are examples of attribute normalization. Given the following declarations:

```
<!ENTITY d "&#xD;">
<!ENTITY a "&#xA;">
<!ENTITY da "&#xD;&#xA;">
```

the attribute specifications in the left column below would be normalized to the character sequences of the middle column if the attribute `a` is declared **NMTOKENS** and to those of the right columns if `a` is declared **CDATA**.

Attribute specification	a is NMTOKENS	a is CDATA

`a="`
 	
`xyz"`
	`x y z`	`#x20 #x20 x y z`

`a="&d;&d;A&a;&#x20;&a;B&da;"`	`A #x20 B`	`#x20 #x20 A #x20 #x20 #x20 B #x20 #x20`

`a=`
`"&#xd;&#xd;A&#xa;&#xa;B&#xd;&#xa;"`
	`#xD #xD A #xA #xA B #xD #xA`	`#xD #xD A #xA #xA B #xD #xA`

Note that the last example is invalid (but well-formed) if `a` is declared to be of type **NMTOKENS**.

3.4 Conditional Sections

[Definition: **Conditional sections** are portions of the document type declaration external subset or of external parameter entities which are included in, or excluded from, the logical structure of the DTD based on the keyword which governs them.]

Conditional Section

[61]	`conditionalSect`	::=	`includeSect	ignoreSect`	
[62]	`includeSect`	::=	`'<![' S? 'INCLUDE' S? '[' extSubsetDecl ']]>'`	[VC: Proper Conditional Section/PE Nesting]
[63]	`ignoreSect`	::=	`'<![' S? 'IGNORE' S? '[' ignoreSectContents* ']]>'`	[VC: Proper Conditional Section/PE Nesting]
[64]	`ignoreSectContents`	::=	`Ignore ('<![' ignoreSectContents ']]>' Ignore)*`	
[65]	`Ignore`	::=	`Char* - (Char* ('<![' | ']]>') Char*)`	

Validity constraint: Proper Conditional Section/PE Nesting

If any of the "`<![`", "`[`", or "`]]>`" of a conditional section is contained in the replacement text for a parameter-entity reference, all of them MUST be contained in the same replacement text.

Like the internal and external DTD subsets, a conditional section may contain one or more complete declarations, comments, processing instructions, or nested conditional sections, intermingled with white space.

If the keyword of the conditional section is **INCLUDE**, then the contents of the conditional section MUST be processed as part of the DTD. If the keyword of the conditional section is **IGNORE**, then the contents of the conditional section MUST NOT be processed as part of the DTD. If a conditional section with a keyword of **INCLUDE** occurs within a larger conditional section with a keyword of **IGNORE**, both the outer and the inner conditional sections MUST be ignored. The contents of an ignored conditional section MUST be parsed by ignoring all characters after the "`[`" following the keyword, except conditional section starts "`<![`" and ends "`]]>`", until the matching conditional section end is found. Parameter entity references MUST NOT be recognized in this process.

If the keyword of the conditional section is a parameter-entity reference, the parameter entity MUST be replaced by its content before the processor decides whether to include or ignore the conditional section.

An example:

```
<!ENTITY % draft 'INCLUDE' >
<!ENTITY % final 'IGNORE' >

<![%draft;[
<!ELEMENT book (comments*, title, body, supplements?)>
]]>
<![%final;[
<!ELEMENT book (title, body, supplements?)>
]]>
```

4 Physical Structures

[Definition: An XML document may consist of one or many storage units. These are called **entities**; they all have **content** and are all (except for the document entity and the external DTD subset) identified by entity **name**.] Each XML document has one entity called the document entity, which serves as the starting point for the XML processor and may contain the whole document.

Entities may be either parsed or unparsed. [Definition: The contents of a **parsed entity** are referred to as its replacement text; this text is considered an integral part of the document.]

[Definition: An **unparsed entity** is a resource whose contents may or may not be text, and if text, may be other than XML. Each unparsed entity has an associated notation, identified by name. Beyond a requirement that an XML processor make the identifiers for the entity and notation available to the application, XML places no constraints on the contents of unparsed entities.]

Parsed entities are invoked by name using entity references; unparsed entities by name, given in the value of **ENTITY** or **ENTITIES** attributes.

[Definition: **General entities** are entities for use within the document content. In this specification, general entities are sometimes referred to with the unqualified term *entity* when this leads to no ambiguity.] [Definition: **Parameter entities** are parsed entities for use within the DTD.] These two types of entities use different forms of reference and are recognized in different contexts. Furthermore, they occupy different namespaces; a parameter entity and a general entity with the same name are two distinct entities.

4.1 Character and Entity References

[Definition: A **character reference** refers to a specific character in the ISO/IEC 10646 character set, for example one not directly accessible from available input devices.]

Character Reference

```
[66]  CharRef  ::=  '&#' [0-9]+ ';'
                  | '&#x' [0-9a-fA-F]+ ';'  [WFC: Legal Character]
```

Well-formedness constraint: Legal Character

Characters referred to using character references MUST match the production for Char.

If the character reference begins with " `&#x` ", the digits and letters up to the terminating `;` provide a hexadecimal representation of the character's code point in ISO/IEC 10646. If it begins just with " `&#` ", the digits up to the terminating `;` provide a decimal representation of the character's code point.

[Definition: An **entity reference** refers to the content of a named entity.] [Definition: References to parsed general entities use ampersand (`&`) and semicolon (`;`) as delimiters.] [Definition: **Parameter-entity references** use percent-sign (`%`) and semicolon (`;`) as delimiters.]

Entity Reference

```
[67]  Reference    ::=  EntityRef | CharRef
[68]  EntityRef    ::=  '&' Name ';'     [WFC: Entity Declared]
                                         [VC: Entity Declared]
                                         [WFC: Parsed Entity]
                                         [WFC: No Recursion]
[69]  PEReference  ::=  '%' Name ';'     [VC: Entity Declared]
                                         [WFC: No Recursion]
                                         [WFC: In DTD]
```

Well-formedness constraint: Entity Declared

In a document without any DTD, a document with only an internal DTD subset which contains no parameter entity references, or a document with " `standalone='yes'` ", for an entity reference that does not occur within the external subset or a parameter entity, the Name given in the entity reference MUST match that in an *entity declaration* that does not occur within the external subset or a parameter entity, except that well-formed documents need not declare any of the following entities: `amp`, `lt`, `gt`, `apos`, `quot`. The declaration of a general entity MUST precede any reference to it which appears in a default value in an attribute-list declaration.

Note that non-validating processors are *not obligated to* read and process entity declarations occurring in parameter entities or in the external subset; for such documents, the rule that an entity must be declared is a well-formedness constraint only if *standalone='yes'*.

Validity constraint: Entity Declared

In a document with an external subset or parameter entity references, if the document is not standalone (either "`standalone='no'`" is specified or there is no standalone declaration), then the Name given in the entity reference MUST match that in an *entity declaration*. For interoperability, valid documents SHOULD declare the entities `amp`, `lt`, `gt`, `apos`, `quot`, in the form specified in **4.6 Predefined Entities**. The declaration of a parameter entity MUST precede any reference to it. Similarly, the declaration of a general entity MUST precede any attribute-list declaration containing a default value with a direct or indirect reference to that general entity.

Well-formedness constraint: Parsed Entity

An entity reference MUST NOT contain the name of an unparsed entity. Unparsed entities may be referred to only in attribute values declared to be of type **ENTITY** or **ENTITIES**.

Well-formedness constraint: No Recursion

A parsed entity MUST NOT contain a recursive reference to itself, either directly or indirectly.

Well-formedness constraint: In DTD

Parameter-entity references MUST NOT appear outside the DTD.

Examples of character and entity references:

```
Type <key>less-than</key> (&#x3C;) to save options.
This document was prepared on &docdate; and
is classified &security-level;.
```

Example of a parameter-entity reference:

```
<!-- declare the parameter entity "ISOLat2"... -->
<!ENTITY % ISOLat2
         SYSTEM "http://www.xml.com/iso/isolat2-xml.entities" >
<!-- ... now reference it. -->
%ISOLat2;
```

4.2 Entity Declarations

[Definition: Entities are declared thus:]

Entity Declaration

[70]	`EntityDecl`	::=	`GEDecl | PEDecl`
[71]	`GEDecl`	::=	`'<!ENTITY' S Name S EntityDef S? '>'`
[72]	`PEDecl`	::=	`'<!ENTITY' S '%' S Name S PEDef S? '>'`
[73]	`EntityDef`	::=	`EntityValue | (ExternalID NDataDecl?)`
[74]	`PEDef`	::=	`EntityValue | ExternalID`

The Name identifies the entity in an entity reference or, in the case of an unparsed entity, in the value of an **ENTITY** or **ENTITIES** attribute. If the same entity is declared more than once, the first declaration encountered is binding; at user option, an XML processor MAY issue a warning if entities are declared multiple times.

4.2.1 Internal Entities

[Definition: If the entity definition is an EntityValue, the defined entity is called an **internal entity**. There is no separate physical storage object, and the content of the entity is given in the declaration.] Note that some processing of entity and character references in the literal entity value may be required to produce the correct replacement text: see **4.5 Construction of Entity Replacement Text**.

An internal entity is a parsed entity.

Example of an internal entity declaration:

```
<!ENTITY Pub-Status "This is a pre-release of the
 specification.">
```

4.2.2 External Entities

[Definition: If the entity is not internal, it is an **external entity**, declared as follows:]

External Entity Declaration

[75]	`ExternalID`	::=	`'SYSTEM' S SystemLiteral`	
			`| 'PUBLIC' S PubidLiteral S SystemLiteral`	
[76]	`NDataDecl`	::=	`S 'NDATA' S Name`	[VC: Notation Declared]

If the NDataDecl is present, this is a general unparsed entity; otherwise it is a parsed entity.

Validity constraint: Notation Declared

The Name MUST match the declared name of a notation.

[Definition: The SystemLiteral is called the entity's **system identifier**. It is meant to be converted to a URI reference (as defined in [IETF RFC 3986]), as part of the process of dereferencing it to obtain input for the XML processor to construct the entity's replacement text.] It is an error for a fragment identifier (beginning with a `#` character) to be part of a system identifier. Unless otherwise provided by information outside the scope of this specification (e.g. a special XML element type defined by a particular DTD, or a processing instruction defined by a particular application specification), relative URIs are relative to the location of the resource within which the entity declaration occurs. This is defined to be the external entity containing the '<' which starts the declaration, at the point when it is parsed as a declaration. A URI might thus be relative to the document entity, to the entity containing the external DTD subset, or to some other external parameter entity. Attempts to retrieve the resource identified by a URI may be redirected at the parser level (for example, in an entity resolver) or below (at the protocol level, for example, via an HTTP `Location:` header). In the absence of additional information outside the scope of this specification within the resource, the base URI of a resource is always the URI of the actual resource returned. In other words, it is the URI of the resource retrieved after all redirection has occurred.

System identifiers (and other XML strings meant to be used as URI references) may contain characters that, according to [IETF RFC 3986], must be escaped before a URI can be used to retrieve the referenced resource. The characters to be escaped are the control characters #x0 to #x1F and #x7F (most of which cannot appear in XML), space #x20, the delimiters '<' #x3C, '>' #x3E and '"' #x22, the *unwise* characters '{' #x7B, '}' #x7D, '|' #x7C, '\' #x5C, '^' #x5E and '`' #x60, as well as all characters above #x7F. Since escaping is not always a fully reversible process, it MUST be performed only when absolutely necessary and as late as possible in a processing chain. In particular, neither the process of converting a relative URI to an absolute one nor the process of passing a URI reference to a process or software component responsible for dereferencing it SHOULD trigger escaping. When escaping does occur, it MUST be performed as follows:

1. Each character to be escaped is represented in UTF-8 [Unicode] as one or more bytes.
2. The resulting bytes are escaped with the URI escaping mechanism (that is, converted to `%` *HH*, where HH is the hexadecimal notation of the byte value).
3. The original character is replaced by the resulting character sequence.

Note:

In a future edition of this specification, the XML Core Working Group intends to replace the preceding paragraph and list of steps with a normative reference to an upcoming revision of IETF RFC 3987, which will define "Legacy Extended IRIs (LEIRIs)". When this revision is available, it is the intent of the XML Core WG to use it to replace language similar to the above in any future revisions of XML-related specifications under its purview.

[Definition: In addition to a system identifier, an external identifier may include a **public identifier**.] An XML processor attempting to retrieve the entity's content may use any combination of the public and system identifiers as well as additional information outside the scope of this specification to try to generate an alternative URI reference. If the processor is unable to do so, it MUST use the URI reference specified in the system literal. Before a match is attempted, all strings of white space in the public identifier MUST be normalized to single space characters (#x20), and leading and trailing white space MUST be removed.

Examples of external entity declarations:

```
<!ENTITY open-hatch
         SYSTEM "http://www.textuality.com/boilerplate/OpenHatch.xml">
<!ENTITY open-hatch
         PUBLIC "-//Textuality//TEXT Standard open-hatch boilerplate//EN"
         "http://www.textuality.com/boilerplate/OpenHatch.xml">
<!ENTITY hatch-pic
         SYSTEM "../grafix/OpenHatch.gif"
         NDATA gif >
```

4.3 Parsed Entities

4.3.1 The Text Declaration

External parsed entities SHOULD each begin with a **text declaration**.

Text Declaration

```
[77]  TextDecl  ::=  '<?xml' VersionInfo? EncodingDecl S? '?>'
```

The text declaration MUST be provided literally, not by reference to a parsed entity. The text declaration MUST NOT appear at any position other than the beginning of an external parsed entity. The text declaration in an external parsed entity is not considered part of its replacement text.

4.3.2 Well-Formed Parsed Entities

The document entity is well-formed if it matches the production labeled document. An external general parsed entity is well-formed if it matches the production labeled extParsedEnt. All external parameter entities are well-formed by definition.

Note:

Only parsed entities that are referenced directly or indirectly within the document are required to be well-formed.

Well-Formed External Parsed Entity

```
[78]  extParsedEnt  ::=  TextDecl? content
```

An internal general parsed entity is well-formed if its replacement text matches the production labeled content. All internal parameter entities are well-formed by definition.

A consequence of well-formedness in general entities is that the logical and physical structures in an XML document are properly nested; no start-tag, end-tag, empty-element tag, element, comment, processing instruction, character reference, or entity reference can begin in one entity and end in another.

4.3.3 Character Encoding in Entities

Each external parsed entity in an XML document may use a different encoding for its characters. All XML processors MUST be able to read entities in both the UTF-8 and UTF-16 encodings. The terms "UTF-8" and "UTF-16" in this specification do not apply to related character encodings, including but not limited to UTF-16BE, UTF-16LE, or CESU-8.

Entities encoded in UTF-16 MUST and entities encoded in UTF-8 MAY begin with the Byte Order Mark described by Annex H of [ISO/IEC 10646:2000], section 16.8 of [Unicode] (the ZERO WIDTH NO-BREAK SPACE character, #xFEFF). This is an encoding signature, not part of either the markup or the character data of the XML document. XML processors MUST be able to use this character to differentiate between UTF-8 and UTF-16 encoded documents.

If the replacement text of an external entity is to begin with the character U+FEFF, and no text declaration is present, then a Byte Order Mark MUST be present, whether the entity is encoded in UTF-8 or UTF-16.

Although an XML processor is required to read only entities in the UTF-8 and UTF-16 encodings, it is recognized that other encodings are used around the world, and it may be desired for XML processors to read entities that use them. In the absence of external character encoding information (such as MIME headers), parsed entities which are stored in an encoding other than UTF-8 or UTF-16 MUST begin with a text declaration (see **4.3.1 The Text Declaration**) containing an encoding declaration:

Encoding Declaration

[80]	`EncodingDecl`	::=	`S 'encoding' Eq ('"' EncName '"' | "'" EncName "'" )`	
[81]	`EncName`	::=	`[A-Za-z] ([A-Za-z0-9._] | '-')*`	*/* Encoding name contains only Latin characters */*

In the document entity, the encoding declaration is part of the XML declaration. The EncName is the name of the encoding used.

In an encoding declaration, the values " `UTF-8` ", " `UTF-16` ", " `ISO-10646-UCS-2` ", and " `ISO-10646-UCS-4` " SHOULD be used for the various encodings and transformations of Unicode / ISO/IEC 10646, the values " `ISO-8859-1` ", " `ISO-8859-2` ", ... " `ISO-8859-` *n* " (where *n* is the part number) SHOULD be used for the parts of ISO 8859, and the values " `ISO-2022-JP` ", " `Shift_JIS` ", and " `EUC-JP` " SHOULD be used for the various encoded forms of JIS X-0208-1997. It is RECOMMENDED that character encodings registered (as *charset*s) with the Internet Assigned Numbers Authority [IANA-CHARSETS], other than those just listed, be referred to using their registered names; other encodings SHOULD use names starting with an "x-" prefix. XML processors SHOULD match character encoding names in a case-insensitive way and SHOULD either interpret an IANA-registered name as the encoding registered at IANA for that name or treat it as unknown (processors are, of course, not required to support all IANA-registered encodings).

In the absence of information provided by an external transport protocol (e.g. HTTP or MIME), it is a fatal error for an entity including an encoding declaration to be presented to the XML processor in an encoding other than that named in the declaration, or for an entity which begins with neither a Byte Order Mark nor an encoding declaration to use an encoding other than UTF-8. Note that since ASCII is a subset of UTF-8, ordinary ASCII entities do not strictly need an encoding declaration.

It is a fatal error for a TextDecl to occur other than at the beginning of an external entity.

It is a fatal error when an XML processor encounters an entity with an encoding that it is unable to process. It is a fatal error if an XML entity is determined (via default, encoding declaration, or higher-level protocol) to be in a certain encoding but contains byte sequences that are not legal in that encoding. Specifically, it is a fatal error if an entity encoded in UTF-8 contains any ill-formed code unit sequences, as defined in section 3.9 of Unicode [Unicode]. Unless an encoding is determined by a higher-level protocol, it is also a fatal error if an XML entity contains no encoding declaration and its content is not legal UTF-8 or UTF-16.

Examples of text declarations containing encoding declarations:

```
<?xml encoding='UTF-8'?>
<?xml encoding='EUC-JP'?>
```

4.4 XML Processor Treatment of Entities and References

The table below summarizes the contexts in which character references, entity references, and invocations of unparsed entities might appear and the REQUIRED behavior of an XML processor in each case. The labels in the leftmost column describe the recognition context:

Reference in Content

as a reference anywhere after the start-tag and before the end-tag of an element; corresponds to the nonterminal content.

Reference in Attribute Value

as a reference within either the value of an attribute in a start-tag, or a default value in an attribute declaration; corresponds to the nonterminal AttValue.

Occurs as Attribute Value

as a Name, not a reference, appearing either as the value of an attribute which has been declared as type **ENTITY**, or as one of the space-separated tokens in the value of an attribute which has been declared as type **ENTITIES**.

Reference in Entity Value

as a reference within a parameter or internal entity's literal entity value in the entity's declaration; corresponds to the nonterminal EntityValue.

Reference in DTD

as a reference within either the internal or external subsets of the DTD, but outside of an EntityValue, AttValue, PI, Comment, SystemLiteral, PubidLiteral, or the contents of an ignored conditional section (see **3.4 Conditional Sections**).

.

	Entity Type				Character
	Parameter	Internal General	External Parsed General	Unparsed	
Reference in Content	*Not recognized*	*Included*	*Included if validating*	*Forbidden*	*Included*
Reference in Attribute Value	*Not recognized*	*Included in literal*	*Forbidden*	*Forbidden*	*Included*
Occurs as Attribute Value	*Not recognized*	*Forbidden*	*Forbidden*	*Notify*	*Not recognized*
Reference in EntityValue	*Included in literal*	*Bypassed*	*Bypassed*	*Error*	*Included*
Reference in DTD	*Included as PE*	*Forbidden*	*Forbidden*	*Forbidden*	*Forbidden*

4.4.1 Not Recognized

Outside the DTD, the `%` character has no special significance; thus, what would be parameter entity references in the DTD are not recognized as markup in content. Similarly, the names of unparsed entities are not recognized except when they appear in the value of an appropriately declared attribute.

4.4.2 Included

[Definition: An entity is **included** when its replacement text is retrieved and processed, in place of the reference itself, as though it were part of the document at the location the reference was recognized.] The replacement text may contain both character data and (except for parameter entities) markup, which MUST be recognized in the usual way. (The string " `AT&amp;T;` " expands to " `AT&T;` " and the remaining ampersand is not recognized as an entity-reference delimiter.) A character reference is **included** when the indicated character is processed in place of the reference itself.

4.4.3 Included If Validating

When an XML processor recognizes a reference to a parsed entity, in order to validate the document, the processor MUST include its replacement text. If the entity is external, and the processor is not attempting to validate the XML document, the processor MAY, but need not, include the entity's replacement text. If a non-validating processor does not include the replacement text, it MUST inform the application that it recognized, but did not read, the entity.

This rule is based on the recognition that the automatic inclusion provided by the SGML and XML entity mechanism, primarily designed to support modularity in authoring, is not necessarily appropriate for other applications, in particular document browsing. Browsers, for example, when encountering an external parsed entity reference, might choose to provide a visual indication of the entity's presence and retrieve it for display only on demand.

4.4.4 Forbidden

The following are forbidden, and constitute fatal errors:

- the appearance of a reference to an unparsed entity, except in the EntityValue in an entity declaration.
- the appearance of any character or general-entity reference in the DTD except within an EntityValue or AttValue.
- a reference to an external entity in an attribute value.

4.4.5 Included in Literal

When an entity reference appears in an attribute value, or a parameter entity reference appears in a literal entity value, its replacement text MUST be processed in place of the reference itself as though it were part of the document at the location the reference was recognized, except that a single or double quote character in the replacement text MUST always be treated as a normal data character and MUST NOT terminate the literal. For example, this is well-formed:

```
<!ENTITY % YN '"Yes"' >
<!ENTITY WhatHeSaid "He said %YN;" >
```

while this is not:

```
<!ENTITY EndAttr "27'" >
<element attribute='a-&EndAttr;>
```

4.4.6 Notify

When the name of an unparsed entity appears as a token in the value of an attribute of declared type **ENTITY** or **ENTITIES**, a validating processor MUST inform the application of the system and public (if any) identifiers for both the entity and its associated notation.

4.4.7 Bypassed

When a general entity reference appears in the EntityValue in an entity declaration, it MUST be bypassed and left as is.

4.4.8 Included as PE

Just as with external parsed entities, parameter entities need only be *included if validating*. When a parameter-entity reference is recognized in the DTD and included, its replacement text MUST be enlarged by the attachment of one leading and one following space (#x20) character; the intent is to constrain the replacement text of parameter entities to contain an integral number of grammatical tokens in the DTD. This behavior MUST NOT apply to parameter entity references within entity values; these are described in **4.4.5 Included in Literal**.

4.4.9 Error

It is an error for a reference to an unparsed entity to appear in the EntityValue in an entity declaration.

4.5 Construction of Entity Replacement Text

In discussing the treatment of entities, it is useful to distinguish two forms of the entity's value. [Definition: For an internal entity, the **literal entity value** is the quoted string actually present in the entity declaration, corresponding to the non-terminal EntityValue.] [Definition: For an external entity, the **literal entity value** is the exact text contained in the entity.] [Definition: For an internal entity, the **replacement text** is the content of the entity, after replacement of character references and parameter-entity references.] [Definition: For an external entity, the **replacement text** is the content of the entity, after stripping the text declaration (leaving any surrounding whitespace) if there is one but without any replacement of character references or parameter-entity references.]

The literal entity value as given in an internal entity declaration (EntityValue) may contain character, parameter-entity, and general-entity references. Such references MUST be contained entirely within the literal entity value. The actual replacement text that is included (or included in literal) as described above MUST contain the *replacement text* of any parameter entities referred to, and MUST contain the character referred to, in place of any character references in the literal entity value; however, general-entity references MUST be left as-is, unexpanded. For example, given the following declarations:

```
<!ENTITY % pub    "&#xc9;ditions Gallimard" >
<!ENTITY   rights "All rights reserved" >
<!ENTITY   book   "La Peste: Albert Camus,
&#xA9; 1947 %pub;. &rights;" >
```

then the replacement text for the entity " `book` " is:

```
La Peste: Albert Camus,
© 1947 Éditions Gallimard. &rights;
```

The general-entity reference " `&rights;` " would be expanded should the reference " `&book;` " appear in the document's content or an attribute value.

These simple rules may have complex interactions; for a detailed discussion of a difficult example, see **D Expansion of Entity and Character References**.

4.6 Predefined Entities

[Definition: Entity and character references may both be used to **escape** the left angle bracket, ampersand, and other delimiters. A set of general entities (`amp`, `lt`, `gt`, `apos`, `quot`) is specified for this purpose. Numeric character references may also be used; they are expanded immediately when recognized and MUST be treated as character data, so the numeric character references " `&#60;` " and " `&#38;` " may be used to escape `<` and `&` when they occur in character data.]

All XML processors MUST recognize these entities whether they are declared or not. For interoperability, valid XML documents SHOULD declare these entities, like any others, before using them. If the entities `lt` or `amp` are declared, they MUST be declared as internal entities whose replacement text is a character reference to the respective character (less-than sign or ampersand) being escaped; the double escaping is REQUIRED for these entities so that references to them produce a well-formed result. If the entities `gt`, `apos`, or `quot` are declared, they MUST be declared as internal entities whose replacement text is the single character being escaped (or a character reference to that character; the double escaping here is OPTIONAL but harmless). For example:

```
<!ENTITY lt     "&#38;#60;">
<!ENTITY gt     "&#62;">
<!ENTITY amp    "&#38;#38;">
<!ENTITY apos   "&#39;">
<!ENTITY quot   "&#34;">
```

4.7 Notation Declarations

[Definition: **Notations** identify by name the format of unparsed entities, the format of elements which bear a notation attribute, or the application to which a processing instruction is addressed.]

[Definition: **Notation declarations** provide a name for the notation, for use in entity and attribute-list declarations and in attribute specifications, and an external identifier for the notation which may allow an XML processor or its client application to locate a helper application capable of processing data in the given notation.]

Notation Declarations

```
[82]  NotationDecl  ::=  '<!NOTATION' S Name S (ExternalID | PublicID) S? '>' [VC: Unique Notation Name]
[83]  PublicID      ::=  'PUBLIC' S PubidLiteral
```

Validity constraint: Unique Notation Name

A given Name MUST NOT be declared in more than one notation declaration.

XML processors MUST provide applications with the name and external identifier(s) of any notation declared and referred to in an attribute value, attribute definition, or entity declaration. They MAY additionally resolve the external identifier into the system identifier, file name, or other information needed to allow the application to call a processor for data in the notation described. (It is not an error, however, for XML documents to declare and refer to notations for which notation-specific applications are not available on the system where the XML processor or application is running.)

4.8 Document Entity

[Definition: The **document entity** serves as the root of the entity tree and a starting-point for an XML processor.] This specification does not specify how the document entity is to be located by an XML processor; unlike other entities, the document entity has no name and might well appear on a processor input stream without any identification at all.

5 Conformance

5.1 Validating and Non-Validating Processors

Conforming XML processors fall into two classes: validating and non-validating.

Validating and non-validating processors alike MUST report violations of this specification's well-formedness constraints in the content of the document entity and any other parsed entities that they read.

[Definition: **Validating processors** MUST, at user option, report violations of the constraints expressed by the declarations in the DTD, and failures to fulfill the validity constraints given in this specification.] To accomplish this, validating XML processors MUST read and process the entire DTD and all external parsed entities referenced in the document.

Non-validating processors are REQUIRED to check only the document entity, including the entire internal DTD subset, for well-formedness. [Definition: While they are not required to check the document for validity, they are REQUIRED to **process** all the declarations they read in the internal DTD subset and in any parameter entity that they read, up to the first reference to a parameter entity that they do *not* read; that is to say, they MUST use the information in those declarations to *normalize* attribute values, *include* the replacement text of internal entities, and supply *default attribute values*.] Except when `standalone="yes"`, they MUST NOT process entity declarations or attribute-list declarations encountered after a reference to a parameter entity that is not read, since the entity may have contained overriding declarations; when `standalone="yes"`, processors MUST process these declarations.

Note that when processing invalid documents with a non-validating processor the application may not be presented with consistent information. For example, several requirements for uniqueness within the document may not be met, including more than one element with the same id, duplicate declarations of elements or notations with the same name, etc. In these cases the behavior of the parser with respect to reporting such information to the application is undefined.

5.2 Using XML Processors

The behavior of a validating XML processor is highly predictable; it must read every piece of a document and report all well-formedness and validity violations. Less is required of a non-validating processor; it need not read any part of the document other than the document entity. This has two effects that may be important to users of XML processors:

- Certain well-formedness errors, specifically those that require reading external entities, may fail to be detected by a non-validating processor. Examples include the constraints entitled *Entity Declared*, *Parsed Entity*, and *No Recursion*, as well as some of the cases described as *forbidden* in **4.4 XML Processor Treatment of Entities and References**.
- The information passed from the processor to the application may vary, depending on whether the processor reads parameter and external entities. For example, a non-validating processor may fail to *normalize* attribute values, *include* the replacement text of internal entities, or supply *default attribute values*, where doing so depends on having read declarations in external or parameter entities, or in the internal subset after an unread parameter entity reference.

For maximum reliability in interoperating between different XML processors, applications which use non-validating processors SHOULD NOT rely on any behaviors not required of such processors. Applications which require DTD facilities not related to validation (such as the declaration of default attributes and internal entities that are or may be specified in external entities) SHOULD use validating XML processors.

6 Notation

The formal grammar of XML is given in this specification using a simple Extended Backus-Naur Form (EBNF) notation. Each rule in the grammar defines one symbol, in the form

```
symbol ::= expression
```

Symbols are written with an initial capital letter if they are the start symbol of a regular language, otherwise with an initial lowercase letter. Literal strings are quoted.

Within the expression on the right-hand side of a rule, the following expressions are used to match strings of one or more characters:

`#xN`

where `N` is a hexadecimal integer, the expression matches the character whose number (code point) in ISO/IEC 10646 is `N`. The number of leading zeros in the `#xN` form is insignificant.

`[a-zA-Z], [#xN-#xN]`

matches any Char with a value in the range(s) indicated (inclusive).

`[abc]`, `[#xN#xN#xN]`

matches any Char with a value among the characters enumerated. Enumerations and ranges can be mixed in one set of brackets.

`[^a-z]`, `[^#xN-#xN]`

matches any Char with a value *outside* the range indicated.

`[^abc]`, `[^#xN#xN#xN]`

matches any Char with a value not among the characters given. Enumerations and ranges of forbidden values can be mixed in one set of brackets.

`"string"`

matches a literal string matching that given inside the double quotes.

`'string'`

matches a literal string matching that given inside the single quotes.

These symbols may be combined to match more complex patterns as follows, where `A` and `B` represent simple expressions:

(`expression`)

`expression` is treated as a unit and may be combined as described in this list.

`A?`

matches `A` or nothing; optional `A`.

`A B`

matches `A` followed by `B`. This operator has higher precedence than alternation; thus `A B | C D` is identical to `(A B) | (C D)`.

`A | B`

matches `A` or `B`.

`A - B`

matches any string that matches `A` but does not match `B`.

`A+`

matches one or more occurrences of `A`. Concatenation has higher precedence than alternation; thus `A+ | B+` is identical to `(A+) | (B+)`.

`A*`

matches zero or more occurrences of `A`. Concatenation has higher precedence than alternation; thus `A* | B*` is identical to `(A*) | (B*)`.

Other notations used in the productions are:

`/* ... */`

comment.

`[ wfc: ... ]`

well-formedness constraint; this identifies by name a constraint on well-formed documents associated with a production.

`[ vc: ... ]`

validity constraint; this identifies by name a constraint on valid documents associated with a production.

A References

A.1 Normative References

IANA-CHARSETS

(Internet Assigned Numbers Authority) *Official Names for Character Sets*, ed. Keld Simonsen et al. (See http://www.iana.org/assignments/character-sets.)

IETF RFC 2119
IETF (Internet Engineering Task Force). *RFC 2119: Key words for use in RFCs to Indicate Requirement Levels*. Scott Bradner, 1997. (See http://www.ietf.org/rfc/rfc2119.txt.)

IETF BCP 47
IETF (Internet Engineering Task Force). *BCP 47, consisting of* RFC 4646: Tags for Identifying Languages, and RFC 4647: Matching of Language Tags, A. Phillips, M. Davis. 2006.

IETF RFC 3986
IETF (Internet Engineering Task Force). *RFC 3986: Uniform Resource Identifier (URI): Generic Syntax*. T. Berners-Lee, R. Fielding, L. Masinter. 2005. (See http://www.ietf.org/rfc/rfc3986.txt.)

ISO/IEC 10646
ISO (International Organization for Standardization). *ISO/IEC 10646-1:2000. Information technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane* and *ISO/IEC 10646-2:2001. Information technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 2: Supplementary Planes*, as, from time to time, amended, replaced by a new edition or expanded by the addition of new parts. [Geneva]: International Organization for Standardization. (See http://www.iso.org/iso/home.htm for the latest version.)

ISO/IEC 10646:2000
ISO (International Organization for Standardization). *ISO/IEC 10646-1:2000. Information technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane.* [Geneva]: International Organization for Standardization, 2000.

Unicode
The Unicode Consortium. *The Unicode Standard, Version 5.0.0,* defined by: The Unicode Standard, Version 5.0 (Boston, MA, Addison-Wesley, 2007. ISBN 0-321-48091-0).

UnicodeNormal
The Unicode Consortium. *Unicode normalization forms*. Mark Davis and Martin Durst. 2008. (See http://unicode.org/reports/tr15/.)

A.2 Other References

Aho/Ullman
Aho, Alfred V., Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Reading: Addison-Wesley, 1986, rpt. corr. 1988.

Brüggemann-Klein
Brüggemann-Klein, Anne. *Formal Models in Document Processing*. Habilitationsschrift. Faculty of Mathematics at the University of Freiburg, 1993. (See ftp://ftp.informatik.uni-freiburg.de/documents/papers/brueggem/habil.ps.)

Brüggemann-Klein and Wood
Brüggemann-Klein, Anne, and Derick Wood. *Deterministic Regular Languages*. Universität Freiburg, Institut für Informatik, Bericht 38, Oktober 1991. Extended abstract in A. Finkel, M. Jantzen, Hrsg., STACS 1992, S. 173-184. Springer-Verlag, Berlin 1992. Lecture Notes in Computer Science 577. Full version titled *One-Unambiguous Regular Languages* in Information and Computation 140 (2): 229-253, February 1998.

Clark
James Clark. *Comparison of SGML and XML*. (See http://www.w3.org/TR/NOTE-sgml-xml-971215.)

IANA-LANGCODES
(Internet Assigned Numbers Authority) *Registry of Language Tags* (See http://www.iana.org/assignments/language-subtag-registry.)

IETF RFC 2141
IETF (Internet Engineering Task Force). *RFC 2141: URN Syntax*, ed. R. Moats. 1997. (See http://www.ietf.org/rfc/rfc2141.txt.)

IETF RFC 3023
IETF (Internet Engineering Task Force). *RFC 3023: XML Media Types*. eds. M. Murata, S. St.Laurent, D. Kohn. 2001. (See http://www.ietf.org/rfc/rfc3023.txt.)

IETF RFC 2781
IETF (Internet Engineering Task Force). *RFC 2781: UTF-16, an encoding of ISO 10646*, ed. P. Hoffman, F. Yergeau. 2000. (See http://www.ietf.org/rfc/rfc2781.txt.)

ISO 639
(International Organization for Standardization). *ISO 639:1988 (E). Code for the representation of names of languages.* [Geneva]: International Organization for Standardization, 1988.

ISO 3166
(International Organization for Standardization). *ISO 3166-1:1997 (E). Codes for the representation of names of countries and their subdivisions — Part 1: Country codes* [Geneva]: International Organization for Standardization, 1997.

ISO 8879
ISO (International Organization for Standardization). *ISO 8879:1986(E). Information processing — Text and Office Systems — Standard Generalized Markup Language (SGML).* First edition — 1986-10-15. [Geneva]: International Organization for Standardization, 1986.

ISO/IEC 10744

ISO (International Organization for Standardization). *ISO/IEC 10744-1992 (E). Information technology — Hypermedia/Time-based Structuring Language (HyTime).* [Geneva]: International Organization for Standardization, 1992. *Extended Facilities Annexe.* [Geneva]: International Organization for Standardization, 1996.

WEBSGML

ISO (International Organization for Standardization). *ISO 8879:1986 TC2. Information technology — Document Description and Processing Languages*. [Geneva]: International Organization for Standardization, 1998. (See http://www.sgmlsource.com/8879/n0029.htm.)

XML Names

Tim Bray, Dave Hollander, and Andrew Layman, editors. *Namespaces in XML*. Textuality, Hewlett-Packard, and Microsoft. World Wide Web Consortium, 1999. (See http://www.w3.org/TR/xml-names/.)

B Character Classes

Because of changes to productions [4] and [5], the productions in this Appendix are now orphaned and not used anymore in determining name characters. This Appendix may be removed in a future edition of this specification; other specifications that wish to refer to the productions herein should do so by means of a reference to the relevant production(s) in the Fourth Edition of this specification.

Following the characteristics defined in the Unicode standard, characters are classed as base characters (among others, these contain the alphabetic characters of the Latin alphabet), ideographic characters, and combining characters (among others, this class contains most diacritics). Digits and extenders are also distinguished.

Characters

```
[84] Letter         ::= BaseChar | Ideographic
[85] BaseChar       ::= [#x0041-#x005A] | [#x0061-#x007A] | [#x00C0-#x00D6] | [#x00D8-#x00F6]
                        | [#x00F8-#x00FF] | [#x0100-#x0131] | [#x0134-#x013E] | [#x0141-#x0148]
                        | [#x014A-#x017E] | [#x0180-#x01C3] | [#x01CD-#x01F0] | [#x01F4-#x01F5]
                        | [#x01FA-#x0217] | [#x0250-#x02A8] | [#x02BB-#x02C1] | #x0386 | [#x0388-#x038A]
                        | #x038C | [#x038E-#x03A1] | [#x03A3-#x03CE] | [#x03D0-#x03D6] | #x03DA | #x03DC
                        | #x03DE | #x03E0 | [#x03E2-#x03F3] | [#x0401-#x040C] | [#x040E-#x044F]
                        | [#x0451-#x045C] | [#x045E-#x0481] | [#x0490-#x04C4] | [#x04C7-#x04C8]
                        | [#x04CB-#x04CC] | [#x04D0-#x04EB] | [#x04EE-#x04F5] | [#x04F8-#x04F9]
                        | [#x0531-#x0556] | #x0559 | [#x0561-#x0586] | [#x05D0-#x05EA] | [#x05F0-#x05F2]
                        | [#x0621-#x063A] | [#x0641-#x064A] | [#x0671-#x06B7] | [#x06BA-#x06BE]
                        | [#x06C0-#x06CE] | [#x06D0-#x06D3] | #x06D5 | [#x06E5-#x06E6] | [#x0905-#x0939]
                        | #x093D | [#x0958-#x0961] | [#x0985-#x098C] | [#x098F-#x0990] | [#x0993-#x09A8]
                        | [#x09AA-#x09B0] | #x09B2 | [#x09B6-#x09B9] | [#x09DC-#x09DD] | [#x09DF-#x09E1]
                        | [#x09F0-#x09F1] | [#x0A05-#x0A0A] | [#x0A0F-#x0A10] | [#x0A13-#x0A28]
                        | [#x0A2A-#x0A30] | [#x0A32-#x0A33] | [#x0A35-#x0A36] | [#x0A38-#x0A39]
                        | [#x0A59-#x0A5C] | #x0A5E | [#x0A72-#x0A74] | [#x0A85-#x0A8B] | #x0A8D
                        | [#x0A8F-#x0A91] | [#x0A93-#x0AA8] | [#x0AAA-#x0AB0] | [#x0AB2-#x0AB3]
                        | [#x0AB5-#x0AB9] | #x0ABD | #x0AE0 | [#x0B05-#x0B0C] | [#x0B0F-#x0B10]
                        | [#x0B13-#x0B28] | [#x0B2A-#x0B30] | [#x0B32-#x0B33] | [#x0B36-#x0B39] | #x0B3D
                        | [#x0B5C-#x0B5D] | [#x0B5F-#x0B61] | [#x0B85-#x0B8A] | [#x0B8E-#x0B90]
                        | [#x0B92-#x0B95] | [#x0B99-#x0B9A] | #x0B9C | [#x0B9E-#x0B9F] | [#x0BA3-#x0BA4]
                        | [#x0BA8-#x0BAA] | [#x0BAE-#x0BB5] | [#x0BB7-#x0BB9] | [#x0C05-#x0C0C]
                        | [#x0C0E-#x0C10] | [#x0C12-#x0C28] | [#x0C2A-#x0C33] | [#x0C35-#x0C39]
                        | [#x0C60-#x0C61] | [#x0C85-#x0C8C] | [#x0C8E-#x0C90] | [#x0C92-#x0CA8]
                        | [#x0CAA-#x0CB3] | [#x0CB5-#x0CB9] | #x0CDE | [#x0CE0-#x0CE1] | [#x0D05-#x0D0C]
                        | [#x0D0E-#x0D10] | [#x0D12-#x0D28] | [#x0D2A-#x0D39] | [#x0D60-#x0D61]
                        | [#x0E01-#x0E2E] | #x0E30 | [#x0E32-#x0E33] | [#x0E40-#x0E45] | [#x0E81-#x0E82]
                        | #x0E84 | [#x0E87-#x0E88] | #x0E8A | #x0E8D | [#x0E94-#x0E97] | [#x0E99-#x0E9F]
                        | [#x0EA1-#x0EA3] | #x0EA5 | #x0EA7 | [#x0EAA-#x0EAB] | [#x0EAD-#x0EAE] | #x0EB0
                        | [#x0EB2-#x0EB3] | #x0EBD | [#x0EC0-#x0EC4] | [#x0F40-#x0F47] | [#x0F49-#x0F69]
                        | [#x10A0-#x10C5] | [#x10D0-#x10F6] | #x1100 | [#x1102-#x1103] | [#x1105-#x1107]
                        | #x1109 | [#x110B-#x110C] | [#x110E-#x1112] | #x113C | #x113E | #x1140 | #x114C
                        | #x114E | #x1150 | [#x1154-#x1155] | #x1159 | [#x115F-#x1161] | #x1163 | #x1165
                        | #x1167 | #x1169 | [#x116D-#x116E] | [#x1172-#x1173] | #x1175 | #x119E | #x11A8
                        | #x11AB | [#x11AE-#x11AF] | [#x11B7-#x11B8] | #x11BA | [#x11BC-#x11C2] | #x11EB
                        | #x11F0 | #x11F9 | [#x1E00-#x1E9B] | [#x1EA0-#x1EF9] | [#x1F00-#x1F15]
                        | [#x1F18-#x1F1D] | [#x1F20-#x1F45] | [#x1F48-#x1F4D] | [#x1F50-#x1F57] | #x1F59
                        | #x1F5B | #x1F5D | [#x1F5F-#x1F7D] | [#x1F80-#x1FB4] | [#x1FB6-#x1FBC] | #x1FBE
                        | [#x1FC2-#x1FC4] | [#x1FC6-#x1FCC] | [#x1FD0-#x1FD3] | [#x1FD6-#x1FDB]
                        | [#x1FE0-#x1FEC] | [#x1FF2-#x1FF4] | [#x1FF6-#x1FFC] | #x2126 | [#x212A-#x212B]
                        | #x212E | [#x2180-#x2182] | [#x3041-#x3094] | [#x30A1-#x30FA] | [#x3105-#x312C]
                        | [#xAC00-#xD7A3]
[86] Ideographic    ::= [#x4E00-#x9FA5] | #x3007 | [#x3021-#x3029]
[87] CombiningChar  ::= [#x0300-#x0345] | [#x0360-#x0361] | [#x0483-#x0486] | [#x0591-#x05A1]
                        | [#x05A3-#x05B9] | [#x05BB-#x05BD] | #x05BF | [#x05C1-#x05C2] | #x05C4
                        | [#x064B-#x0652] | #x0670 | [#x06D6-#x06DC] | [#x06DD-#x06DF] | [#x06E0-#x06E4]
                        | [#x06E7-#x06E8] | [#x06EA-#x06ED] | [#x0901-#x0903] | #x093C | [#x093E-#x094C]
                        | #x094D | [#x0951-#x0954] | [#x0962-#x0963] | [#x0981-#x0983] | #x09BC | #x09BE
                        | #x09BF | [#x09C0-#x09C4] | [#x09C7-#x09C8] | [#x09CB-#x09CD] | #x09D7
                        | [#x09E2-#x09E3] | #x0A02 | #x0A3C | #x0A3E | #x0A3F | [#x0A40-#x0A42]
                        | [#x0A47-#x0A48] | [#x0A4B-#x0A4D] | [#x0A70-#x0A71] | [#x0A81-#x0A83] | #x0ABC
                        | [#x0ABE-#x0AC5] | [#x0AC7-#x0AC9] | [#x0ACB-#x0ACD] | [#x0B01-#x0B03] | #x0B3C
```

```
                              | [#x0B3E-#x0B43] | [#x0B47-#x0B48] | [#x0B4B-#x0B4D] | [#x0B56-#x0B57]
                              | [#x0B82-#x0B83] | [#x0BBE-#x0BC2] | [#x0BC6-#x0BC8] | [#x0BCA-#x0BCD] | #x0BD7
                              | [#x0C01-#x0C03] | [#x0C3E-#x0C44] | [#x0C46-#x0C48] | [#x0C4A-#x0C4D]
                              | [#x0C55-#x0C56] | [#x0C82-#x0C83] | [#x0CBE-#x0CC4] | [#x0CC6-#x0CC8]
                              | [#x0CCA-#x0CCD] | [#x0CD5-#x0CD6] | [#x0D02-#x0D03] | [#x0D3E-#x0D43]
                              | [#x0D46-#x0D48] | [#x0D4A-#x0D4D] | #x0D57 | #x0E31 | [#x0E34-#x0E3A]
                              | [#x0E47-#x0E4E] | #x0EB1 | [#x0EB4-#x0EB9] | [#x0EBB-#x0EBC] | [#x0EC8-#x0ECD]
                              | [#x0F18-#x0F19] | #x0F35 | #x0F37 | #x0F39 | #x0F3E | #x0F3F | [#x0F71-#x0F84]
                              | [#x0F86-#x0F8B] | [#x0F90-#x0F95] | #x0F97 | [#x0F99-#x0FAD] | [#x0FB1-#x0FB7]
                              | #x0FB9 | [#x20D0-#x20DC] | #x20E1 | [#x302A-#x302F] | #x3099 | #x309A
[88]  Digit             ::=   [#x0030-#x0039] | [#x0660-#x0669] | [#x06F0-#x06F9] | [#x0966-#x096F]
                              | [#x09E6-#x09EF] | [#x0A66-#x0A6F] | [#x0AE6-#x0AEF] | [#x0B66-#x0B6F]
                              | [#x0BE7-#x0BEF] | [#x0C66-#x0C6F] | [#x0CE6-#x0CEF] | [#x0D66-#x0D6F]
                              | [#x0E50-#x0E59] | [#x0ED0-#x0ED9] | [#x0F20-#x0F29]
[89]  Extender          ::=   #x00B7 | #x02D0 | #x02D1 | #x0387 | #x0640 | #x0E46 | #x0EC6 | #x3005
                              | [#x3031-#x3035] | [#x309D-#x309E] | [#x30FC-#x30FE]
```

The character classes defined here can be derived from the Unicode 2.0 character database as follows:

- Name start characters must have one of the categories Ll, Lu, Lo, Lt, Nl.
- Name characters other than Name-start characters must have one of the categories Mc, Me, Mn, Lm, or Nd.
- Characters in the compatibility area (i.e. with character code greater than #xF900 and less than #xFFFE) are not allowed in XML names.
- Characters which have a font or compatibility decomposition (i.e. those with a "compatibility formatting tag" in field 5 of the database -- marked by field 5 beginning with a "<") are not allowed.
- The following characters are treated as name-start characters rather than name characters, because the property file classifies them as Alphabetic: [#x02BB-#x02C1], #x0559, #x06E5, #x06E6.
- Characters #x20DD-#x20E0 are excluded (in accordance with Unicode 2.0, section 5.14).
- Character #x00B7 is classified as an extender, because the property list so identifies it.
- Character #x0387 is added as a name character, because #x00B7 is its canonical equivalent.
- Characters ':' and '_' are allowed as name-start characters.
- Characters '-' and '.' are allowed as name characters.

C XML and SGML (Non-Normative)

XML is designed to be a subset of SGML, in that every XML document should also be a conforming SGML document. For a detailed comparison of the additional restrictions that XML places on documents beyond those of SGML, see [Clark].

D Expansion of Entity and Character References (Non-Normative)

This appendix contains some examples illustrating the sequence of entity- and character-reference recognition and expansion, as specified in **4.4 XML Processor Treatment of Entities and References**.

If the DTD contains the declaration

```
<!ENTITY example "<p>An ampersand (&#38;#38;) may be escaped
numerically (&#38;#38;#38;) or with a general entity
(&amp;amp;).</p>" >
```

then the XML processor will recognize the character references when it parses the entity declaration, and resolve them before storing the following string as the value of the entity " `example` ":

```
<p>An ampersand (&#38;) may be escaped
numerically (&#38;#38;) or with a general entity
(&amp;amp;).</p>
```

A reference in the document to " `&example;` " will cause the text to be reparsed, at which time the start- and end-tags of the `p` element will be recognized and the three references will be recognized and expanded, resulting in a `p` element with the following content (all data, no delimiters or markup):

```
An ampersand (&) may be escaped
numerically (&#38;) or with a general entity
(&amp;).
```

A more complex example will illustrate the rules and their effects fully. In the following example, the line numbers are solely for reference.

```
1 <?xml version='1.0'?>
2 <!DOCTYPE test [
3 <!ELEMENT test (#PCDATA) >
4 <!ENTITY % xx '&#37;zz;'>
5 <!ENTITY % zz '&#60;!ENTITY tricky "error-prone" >' >
6 %xx;
7 ]>
8 <test>This sample shows a &tricky; method.</test>
```

This produces the following:

- in line 4, the reference to character 37 is expanded immediately, and the parameter entity " `xx` " is stored in the symbol table with the value " `%zz;` ". Since the replacement text is not rescanned, the reference to parameter entity " `zz` " is not recognized. (And it would be an error if it were, since " `zz` " is not yet declared.)
- in line 5, the character reference " `&#60;` " is expanded immediately and the parameter entity " `zz` " is stored with the replacement text " `<!ENTITY tricky "error-prone" >` ", which is a well-formed entity declaration.
- in line 6, the reference to " `xx` " is recognized, and the replacement text of " `xx` " (namely " `%zz;` ") is parsed. The reference to " `zz` " is recognized in its turn, and its replacement text (" `<!ENTITY tricky "error-prone">` ") is parsed. The general entity " `tricky` " has now been declared, with the replacement text " `error-prone` ".
- in line 8, the reference to the general entity " `tricky` " is recognized, and it is expanded, so the full content of the `test` element is the self-describing (and ungrammatical) string *This sample shows a error-prone method.*

In the following example

```
<!DOCTYPE foo [
<!ENTITY x "&lt;">
]>
<foo attr="&x;"/>
```

the replacement text of x is the four characters "&lt;" because references to general entities in entity values are *bypassed*. The replacement text of lt is a character reference to the less-than character, for example the five characters "&#60;" (see **4.6 Predefined Entities**). Since neither of these contains a less-than character the result is well-formed.

If the definition of x had been

```
<!ENTITY x "&#60;">
```

then the document would not have been well-formed, because the replacement text of x would be the single character "<" which is not permitted in attribute values (see *WFC: No < in Attribute Values*).

E Deterministic Content Models (Non-Normative)

As noted in **3.2.1 Element Content**, it is required that content models in element type declarations be deterministic. This requirement is for compatibility with SGML (which calls deterministic content models "unambiguous"); XML processors built using SGML systems may flag non-deterministic content models as errors.

For example, the content model `((b, c) | (b, d))` is non-deterministic, because given an initial `b` the XML processor cannot know which `b` in the model is being matched without looking ahead to see which element follows the `b`. In this case, the two references to `b` can be collapsed into a single reference, making the model read `(b, (c | d))`. An initial `b` now clearly matches only a single name in the content model. The processor doesn't need to look ahead to see what follows; either `c` or `d` would be accepted.

More formally: a finite state automaton may be constructed from the content model using the standard algorithms, e.g. algorithm 3.5 in section 3.9 of Aho, Sethi, and Ullman [Aho/Ullman]. In many such algorithms, a follow set is constructed for each position in the regular expression (i.e., each leaf node in the syntax tree for the regular expression); if any position has a follow set in which more than one following position is labeled with the same element type name, then the content model is in error and may be reported as an error.

Algorithms exist which allow many but not all non-deterministic content models to be reduced automatically to equivalent deterministic models; see Brüggemann-Klein 1991 [Brüggemann-Klein].

F Autodetection of Character Encodings (Non-Normative)

The XML encoding declaration functions as an internal label on each entity, indicating which character encoding is in use. Before an XML processor can read the internal label, however, it apparently has to know what character encoding is in use—which is what the internal label is trying to indicate. In the general case, this is a hopeless situation. It is not entirely hopeless in XML, however, because XML limits the general case in two ways: each implementation is assumed to support only a finite set of character encodings, and the XML encoding declaration is restricted in position and content in order to make it feasible to autodetect the character encoding in use in each entity in normal cases. Also, in many cases other sources of information are available in addition to the XML data stream itself. Two cases may be distinguished, depending on whether the XML entity is presented to the processor without, or with, any accompanying (external) information. We will consider these cases in turn.

F.1 Detection Without External Encoding Information

Because each XML entity not accompanied by external encoding information and not in UTF-8 or UTF-16 encoding must begin with an XML encoding declaration, in which the first characters must be '`<?xml`', any conforming processor can detect, after two to four octets of input, which of the following cases apply. In reading this list, it may help to know that in UCS-4, '<' is " `#x0000003C` " and '?' is " `#x0000003F` ", and the Byte Order Mark required of UTF-16 data streams is " `#xFEFF` ". The notation ## is used to denote any byte value except that two consecutive ##s cannot be both 00.

With a Byte Order Mark:

`00 00 FE FF`	UCS-4, big-endian machine (1234 order)
`FF FE 00 00`	UCS-4, little-endian machine (4321 order)
`00 00 FF FE`	UCS-4, unusual octet order (2143)
`FE FF 00 00`	UCS-4, unusual octet order (3412)
`FE FF ## ##`	UTF-16, big-endian
`FF FE ## ##`	UTF-16, little-endian
`EF BB BF`	UTF-8

Without a Byte Order Mark:

`00 00 00 3C`
`3C 00 00 00`
`00 00 3C 00`
`00 3C 00 00`
	UCS-4 or other encoding with a 32-bit code unit and ASCII characters encoded as ASCII values, in respectively big-endian (1234), little-endian (4321) and two unusual byte orders (2143 and 3412). The encoding declaration must be read to determine which of UCS-4 or other supported 32-bit encodings applies.

`00 3C 00 3F`	UTF-16BE or big-endian ISO-10646-UCS-2 or other encoding with a 16-bit code unit in big-endian order and ASCII characters encoded as ASCII values (the encoding declaration must be read to determine which)
`3C 00 3F 00`	UTF-16LE or little-endian ISO-10646-UCS-2 or other encoding with a 16-bit code unit in little-endian order and ASCII characters encoded as ASCII values (the encoding declaration must be read to determine which)
`3C 3F 78 6D`	UTF-8, ISO 646, ASCII, some part of ISO 8859, Shift-JIS, EUC, or any other 7-bit, 8-bit, or mixed-width encoding which ensures that the characters of ASCII have their normal positions, width, and values; the actual encoding declaration must be read to detect which of these applies, but since all of these encodings use the same bit patterns for the relevant ASCII characters, the encoding declaration itself may be read reliably
`4C 6F A7 94`	EBCDIC (in some flavor; the full encoding declaration must be read to tell which code page is in use)
Other	UTF-8 without an encoding declaration, or else the data stream is mislabeled (lacking a required encoding declaration), corrupt, fragmentary, or enclosed in a wrapper of some kind

Note:

In cases above which do not require reading the encoding declaration to determine the encoding, section 4.3.3 still requires that the encoding declaration, if present, be read and that the encoding name be checked to match the actual encoding of the entity. Also, it is possible that new character encodings will be invented that will make it necessary to use the encoding declaration to determine the encoding, in cases where this is not required at present.

This level of autodetection is enough to read the XML encoding declaration and parse the character-encoding identifier, which is still necessary to distinguish the individual members of each family of encodings (e.g. to tell UTF-8 from 8859, and the parts of 8859 from each other, or to distinguish the specific EBCDIC code page in use, and so on).

Because the contents of the encoding declaration are restricted to characters from the ASCII repertoire (however encoded), a processor can reliably read the entire encoding declaration as soon as it has detected which family of encodings is in use. Since in practice, all widely used character encodings fall into one of the categories above, the XML encoding declaration allows reasonably reliable in-band labeling of character encodings, even when external sources of information at the operating-system or transport-protocol level are unreliable. Character encodings such as UTF-7 that make overloaded usage of ASCII-valued bytes may fail to be reliably detected.

Once the processor has detected the character encoding in use, it can act appropriately, whether by invoking a separate input routine for each case, or by calling the proper conversion function on each character of input.

Like any self-labeling system, the XML encoding declaration will not work if any software changes the entity's character set or encoding without updating the encoding declaration. Implementors of character-encoding routines should be careful to ensure the accuracy of the internal and external information used to label the entity.

F.2 Priorities in the Presence of External Encoding Information

The second possible case occurs when the XML entity is accompanied by encoding information, as in some file systems and some network protocols. When multiple sources of information are available, their relative priority and the preferred method of handling conflict should be specified as part of the higher-level protocol used to deliver XML. In particular, please refer to [IETF RFC 3023] or its successor, which defines the `text/xml` and `application/xml` MIME types and provides some useful guidance. In the interests of interoperability, however, the following rule is recommended.

- If an XML entity is in a file, the Byte-Order Mark and encoding declaration are used (if present) to determine the character encoding.

G W3C XML Working Group (Non-Normative)

This specification was prepared and approved for publication by the W3C XML Working Group (WG). WG approval of this specification does not necessarily imply that all WG members voted for its approval. The current and former participants of the XML WG are:

- Jon Bosak, Sun (*Chair*)
- James Clark (*Technical Lead*)
- Tim Bray, Textuality and Netscape (*XML Co-editor*)
- Jean Paoli, Microsoft (*XML Co-editor*)
- C. M. Sperberg-McQueen, U. of Ill. (*XML Co-editor*)
- Dan Connolly, W3C (*W3C Liaison*)
- Paula Angerstein, Texcel
- Steve DeRose, INSO
- Dave Hollander, HP
- Eliot Kimber, ISOGEN
- Eve Maler, ArborText
- Tom Magliery, NCSA
- Murray Maloney, SoftQuad, Grif SA, Muzmo and Veo Systems
- MURATA Makoto (FAMILY Given), Fuji Xerox Information Systems
- Joel Nava, Adobe
- Conleth O'Connell, Vignette
- Peter Sharpe, SoftQuad
- John Tigue, DataChannel

H W3C XML Core Working Group (Non-Normative)

The fifth edition of this specification was prepared by the W3C XML Core Working Group (WG). The participants in the WG at the time of publication of this edition were:

- John Cowan, Google
- Andrew Fang, PTC-Arbortext
- Paul Grosso, PTC-Arbortext (*Co-Chair*)
- Konrad Lanz, A-SIT
- Glenn Marcy, IBM
- Henry Thompson, W3C (*Staff Contact*)
- Richard Tobin, University of Edinburgh
- Daniel Veillard
- Norman Walsh, Mark Logic (*Co-Chair*)
- François Yergeau

I Production Notes (Non-Normative)

This edition was encoded in a slightly modified version of the XMLspec DTD, v2.10. The XHTML versions were produced with a combination of the xmlspec.xsl, diffspec.xsl, and REC-xml.xsl XSLT stylesheets.

J Suggestions for XML Names (Non-Normative)

The following suggestions define what is believed to be best practice in the construction of XML names used as element names, attribute names, processing instruction targets, entity names, notation names, and the values of attributes of type ID, and are intended as guidance for document authors and schema designers. All references to Unicode are understood with respect to a particular version of the Unicode Standard greater than or equal to 5.0; which version should be used is left to the discretion of the document author or schema designer.

The first two suggestions are directly derived from the rules given for identifiers in Standard Annex #31 (UAX #31) of the Unicode Standard, version 5.0 [Unicode], and exclude all control characters, enclosing nonspacing marks, non-decimal numbers, private-use characters, punctuation characters (with the noted exceptions), symbol characters, unassigned codepoints, and white space characters. The other suggestions are mostly derived from Appendix B in previous editions of this specification.

1. The first character of any name should have a Unicode property of ID_Start, or else be '_' #x5F.
2. Characters other than the first should have a Unicode property of ID_Continue, or be one of the characters listed in the table entitled "Characters for Natural Language Identifiers" in UAX #31, with the exception of "'" #x27 and "’" #x2019.
3. Characters in names should be expressed using Normalization Form C as defined in [UnicodeNormal].
4. Ideographic characters which have a canonical decomposition (including those in the ranges [#xF900-#xFAFF] and [#x2F800-#x2FFFD], with 12 exceptions) should not be used in names.
5. Characters which have a compatibility decomposition (those with a "compatibility formatting tag" in field 5 of the Unicode Character Database -- marked by field 5 beginning with a "<") should not be used in names. This suggestion does not apply to characters which despite their compatibility decompositions are in regular use in their scripts, for example #x0E33 THAI CHARACTER SARA AM or #x0EB3 LAO CHARACTER AM.
6. Combining characters meant for use with symbols only (including those in the ranges [#x20D0-#x20EF] and [#x1D165-#x1D1AD]) should not be used in names.
7. The interlinear annotation characters ([#xFFF9-#xFFFB]) should not be used in names.
8. Variation selector characters should not be used in names.
9. Names which are nonsensical, unpronounceable, hard to read, or easily confusable with other names should not be employed.

XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition)

A Reformulation of HTML 4 in XML 1.0

W3C Recommendation 26 January 2000, revised 1 August 2002

This version:
http://www.w3.org/TR/2002/REC-xhtml1-20020801
Latest version:
http://www.w3.org/TR/xhtml1
Previous version:
http://www.w3.org/TR/2000/REC-xhtml1-20000126
Diff-marked version:
http://www.w3.org/TR/2002/REC-xhtml1-20020801/xhtml1-diff.html
Authors:
See acknowledgments.

Please refer to the **errata** for this document, which may include some normative corrections. See also **translations**.

This document is also available in these non-normative formats: Multi-part XHTML file, PostScript version, PDF version, ZIP archive, and Gzip'd TAR archive.

Abstract

This specification defines the Second Edition of XHTML 1.0, a reformulation of HTML 4 as an XML 1.0 application, and three DTDs corresponding to the ones defined by HTML 4. The semantics of the elements and their attributes are defined in the W3C Recommendation for HTML 4. These semantics provide the foundation for future extensibility of XHTML. Compatibility with existing HTML user agents is possible by following a small set of guidelines.

Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained at the W3C.

This document is the second edition of the XHTML 1.0 specification incorporating the errata changes as of 1 August 2002. Changes between this version and the previous Recommendation are illustrated in a diff-marked version.

This second edition is *not* a new version of XHTML 1.0 (first published 26 January 2000). The changes in this document reflect corrections applied as a result of comments submitted by the community and as a result of ongoing work within the HTML Working Group. There are no substantive changes in this document - only the integration of various errata.

The list of known errors in this specification is available at http://www.w3.org/2002/08/REC-xhtml1-20020801-errata.

Please report errors in this document to www-html-editor@w3.org (archive). Public discussion on HTML features takes place on the mailing list www-html@w3.org (archive).

This document has been produced as part of the W3C HTML Activity. The goals of the HTML Working Group *(members only)* are discussed in the HTML Working Group charter.

At the time of publication, the working group believed there were zero patent disclosures relevant to this specification. A current list of patent disclosures relevant to this specification may be found on the Working Group's patent disclosure page.

A list of current W3C Recommendations and other technical documents can be found at http://www.w3.org/TR.

Quick Table of Contents

Full Table of Contents

1. What is XHTML?

This section is informative.

XHTML is a family of current and future document types and modules that reproduce, subset, and extend HTML 4 [HTML4]. XHTML family document types are XML based, and ultimately are designed to work in conjunction with XML-based user agents. The details of this family and its evolution are discussed in more detail in [XHTMLMOD].

XHTML 1.0 (this specification) is the first document type in the XHTML family. It is a reformulation of the three HTML 4 document types as applications of XML 1.0 [XML]. It is intended to be used as a language for content that is both XML-conforming and, if some simple guidelines are followed, operates in HTML 4 conforming user agents. Developers who migrate their content to XHTML 1.0 will realize the following benefits:

- XHTML documents are XML conforming. As such, they are readily viewed, edited, and validated with standard XML tools.
- XHTML documents can be written to operate as well or better than they did before in existing HTML 4-conforming user agents as well as in new, XHTML 1.0 conforming user agents.
- XHTML documents can utilize applications (e.g. scripts and applets) that rely upon either the HTML Document Object Model or the XML Document Object Model [DOM].
- As the XHTML family evolves, documents conforming to XHTML 1.0 will be more likely to interoperate within and among various XHTML environments.

The XHTML family is the next step in the evolution of the Internet. By migrating to XHTML today, content developers can enter the XML world with all of its attendant benefits, while still remaining confident in their content's backward and future compatibility.

1.1. What is HTML 4?

HTML 4 [HTML4] is an SGML (Standard Generalized Markup Language) application conforming to International Standard ISO 8879, and is widely regarded as the standard publishing language of the World Wide Web.

SGML is a language for describing markup languages, particularly those used in electronic document exchange, document management, and document publishing. HTML is an example of a language defined in SGML.

SGML has been around since the middle 1980's and has remained quite stable. Much of this stability stems from the fact that the language is both feature-rich and flexible. This flexibility, however, comes at a price, and that price is a level of complexity that has inhibited its adoption in a diversity of environments, including the World Wide Web.

HTML, as originally conceived, was to be a language for the exchange of scientific and other technical documents, suitable for use by non-document specialists. HTML addressed the problem of SGML complexity by specifying a small set of structural and semantic tags suitable for authoring relatively simple documents. In addition to simplifying the document structure, HTML added support for hypertext. Multimedia capabilities were added later.

In a remarkably short space of time, HTML became wildly popular and rapidly outgrew its original purpose. Since HTML's inception, there has been rapid invention of new elements for use within HTML (as a standard) and for adapting HTML to vertical, highly specialized, markets. This plethora of new elements has led to interoperability problems for documents across different platforms.

1.2. What is XML?

XML™ is the shorthand name for Extensible Markup Language [XML].

XML was conceived as a means of regaining the power and flexibility of SGML without most of its complexity. Although a restricted form of SGML, XML nonetheless preserves most of SGML's power and richness, and yet still retains all of SGML's commonly used features.

While retaining these beneficial features, XML removes many of the more complex features of SGML that make the authoring and design of suitable software both difficult and costly.

1.3. Why the need for XHTML?

The benefits of migrating to XHTML 1.0 are described above. Some of the benefits of migrating to XHTML in general are:

- Document developers and user agent designers are constantly discovering new ways to express their ideas through new markup. In XML, it is relatively easy to introduce new elements or additional element attributes. The XHTML family is designed to accommodate these extensions through XHTML modules and techniques for developing new XHTML-conforming modules (described in the XHTML Modularization specification). These modules will permit the combination of existing and new feature sets when developing content and when designing new user agents.
- Alternate ways of accessing the Internet are constantly being introduced. The XHTML family is designed with general user agent interoperability in mind. Through a new user agent and document profiling mechanism, servers, proxies, and user agents will be able to perform best effort content transformation. Ultimately, it will be possible to develop XHTML-conforming content that is usable by any XHTML-conforming user agent.

2. Definitions

This section is normative.

2.1. Terminology

The following terms are used in this specification. These terms extend the definitions in [RFC2119] in ways based upon similar definitions in ISO/IEC 9945-1:1990 [POSIX.1]:

May
: With respect to implementations, the word "may" is to be interpreted as an optional feature that is not required in this specification but can be provided. With respect to Document Conformance, the word "may" means that the optional feature must not be used. The term "optional" has the same definition as "may".

Must

In this specification, the word "must" is to be interpreted as a mandatory requirement on the implementation or on Strictly Conforming XHTML Documents, depending upon the context. The term "shall" has the same definition as "must".

Optional
: See "May".

Reserved
: A value or behavior is unspecified, but it is not allowed to be used by Conforming Documents nor to be supported by Conforming User Agents.

Shall
: See "Must".

Should
: With respect to implementations, the word "should" is to be interpreted as an implementation recommendation, but not a requirement. With respect to documents, the word "should" is to be interpreted as recommended programming practice for documents and a requirement for Strictly Conforming XHTML Documents.

Supported
: Certain facilities in this specification are optional. If a facility is supported, it behaves as specified by this specification.

Unspecified
: When a value or behavior is unspecified, the specification defines no portability requirements for a facility on an implementation even when faced with a document that uses the facility. A document that requires specific behavior in such an instance, rather than tolerating any behavior when using that facility, is not a Strictly Conforming XHTML Document.

2.2. General Terms

Attribute
: An attribute is a parameter to an element declared in the DTD. An attribute's type and value range, including a possible default value, are defined in the DTD.

DTD
: A DTD, or document type definition, is a collection of XML markup declarations that, as a collection, defines the legal structure, elements, and attributes that are available for use in a document that complies to the DTD.

Document
: A document is a stream of data that, after being combined with any other streams it references, is structured such that it holds information contained within elements that are organized as defined in the associated DTD. See Document Conformance for more information.

Element
: An element is a document structuring unit declared in the DTD. The element's content model is defined in the DTD, and additional semantics may be defined in the prose description of the element.

Facilities
: Facilities are elements, attributes, and the semantics associated with those elements and attributes.

Implementation
: See User Agent.

Parsing
: Parsing is the act whereby a document is scanned, and the information contained within the document is filtered into the context of the elements in which the information is structured.

Rendering

Rendering is the act whereby the information in a document is presented. This presentation is done in the form most appropriate to the environment (e.g. aurally, visually, in print).

User Agent

A user agent is a system that processes XHTML documents in accordance with this specification. See User Agent Conformance for more information.

Validation

Validation is a process whereby documents are verified against the associated DTD, ensuring that the structure, use of elements, and use of attributes are consistent with the definitions in the DTD.

Well-formed

A document is well-formed when it is structured according to the rules defined in Section 2.1 of the XML 1.0 Recommendation [XML].

3. Normative Definition of XHTML 1.0

This section is normative.

3.1. Document Conformance

This version of XHTML provides a definition of strictly conforming XHTML 1.0 documents, which are restricted to elements and attributes from the XML and XHTML 1.0 namespaces. See Section 3.1.2 for information on using XHTML with other namespaces, for instance, to include metadata expressed in RDF within XHTML documents.

3.1.1. Strictly Conforming Documents

A Strictly Conforming XHTML Document is an XML document that requires only the facilities described as mandatory in this specification. Such a document must meet all of the following criteria:

1. It must conform to the constraints expressed in one of the three DTDs found in DTDs and in Appendix B.

2. The root element of the document must be `html`.

3. The root element of the document must contain an `xmlns` declaration for the XHTML namespace [XMLNS]. The namespace for XHTML is defined to be `http://www.w3.org/1999/xhtml`. An example root element might look like:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

4. There must be a DOCTYPE declaration in the document prior to the root element. The public identifier included in the DOCTYPE declaration must reference one of the three DTDs found in DTDs using the respective Formal Public Identifier. The system identifier may be changed to reflect local system conventions.

```
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

5. The DTD subset must not be used to override any parameter entities in the DTD.

An XML declaration is not required in all XML documents; however XHTML document authors are strongly encouraged to use XML declarations in all their documents. Such a declaration is required when the character encoding of the document is other than the default UTF-8 or UTF-16 and no encoding was determined by a higher-level protocol. Here is an example of an XHTML document. In this example, the XML declaration is included.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Virtual Library</title>
  </head>
  <body>
    <p>Moved to <a href="http://example.org/">example.org</a>.</p>
  </body>
</html>
```

3.1.2. Using XHTML with other namespaces

The XHTML namespace may be used with other XML namespaces as per [XMLNS], although such documents are not strictly conforming XHTML 1.0 documents as defined above. Work by W3C is addressing ways to specify conformance for documents involving multiple namespaces. For an example, see [XHTML+MathML].

The following example shows the way in which XHTML 1.0 could be used in conjunction with the MathML Recommendation:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>A Math Example</title>
  </head>
  <body>
    <p>The following is MathML markup:</p>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply> <log/>
        <logbase>
          <cn> 3 </cn>
        </logbase>
        <ci> x </ci>
      </apply>
    </math>
  </body>
</html>
```

The following example shows the way in which XHTML 1.0 markup could be incorporated into another XML namespace:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- initially, the default namespace is "books" -->
```

```
<book xmlns='urn:loc.gov:books'
    xmlns:isbn='urn:ISBN:0-395-36341-6' xml:lang="en" lang="en">
  <title>Cheaper by the Dozen</title>
  <isbn:number>1568491379</isbn:number>
  <notes>
    <!-- make HTML the default namespace for a hypertext commentary -->
    <p xmlns='http://www.w3.org/1999/xhtml'>
        This is also available <a href="http://www.w3.org/">online</a>.
    </p>
  </notes>
</book>
```

3.2. User Agent Conformance

A conforming user agent must meet all of the following criteria:

1. In order to be consistent with the XML 1.0 Recommendation [XML], the user agent must parse and evaluate an XHTML document for well-formedness. If the user agent claims to be a validating user agent, it must also validate documents against their referenced DTDs according to [XML].
2. When the user agent claims to support facilities defined within this specification or required by this specification through normative reference, it must do so in ways consistent with the facilities' definition.
3. When a user agent processes an XHTML document as generic XML, it shall only recognize attributes of type `ID` (i.e. the `id` attribute on most XHTML elements) as fragment identifiers.
4. If a user agent encounters an element it does not recognize, it must process the element's content.
5. If a user agent encounters an attribute it does not recognize, it must ignore the entire attribute specification (i.e., the attribute and its value).
6. If a user agent encounters an attribute value it does not recognize, it must use the default attribute value.
7. If it encounters an entity reference (other than one of the entities defined in this recommendation or in the XML recommendation) for which the user agent has processed no declaration (which could happen if the declaration is in the external subset which the user agent hasn't read), the entity reference should be processed as the characters (starting with the ampersand and ending with the semi-colon) that make up the entity reference.
8. When processing content, user agents that encounter characters or character entity references that are recognized but not renderable may substitute another rendering that gives the same meaning, or must display the document in such a way that it is obvious to the user that normal rendering has not taken place.

9. White space is handled according to the following rules. The following characters are defined in [XML] white space characters:

   - SPACE (&#x0020;)
   - HORIZONTAL TABULATION (&#x0009;)
   - CARRIAGE RETURN (&#x000D;)
   - LINE FEED (&#x000A;)

   The XML processor normalizes different systems' line end codes into one single LINE FEED character, that is passed up to the application.

   The user agent must use the definition from CSS for processing whitespace characters [CSS2]. *Note that the CSS2 recommendation does not explicitly address the issue of whitespace handling in non-*

Latin character sets. This will be addressed in a future version of CSS, at which time this reference will be updated.

Note that in order to produce a Canonical XHTML document, the rules above must be applied and the rules in [XMLC14N] must also be applied to the document.

4. Differences with HTML 4

This section is informative.

Due to the fact that XHTML is an XML application, certain practices that were perfectly legal in SGML-based HTML 4 [HTML4] must be changed.

4.1. Documents must be well-formed

Well-formedness is a new concept introduced by [XML]. Essentially this means that all elements must either have closing tags or be written in a special form (as described below), and that all the elements must nest properly.

Although overlapping is illegal in SGML, it is widely tolerated in existing browsers.

CORRECT: nested elements.

<p>here is an emphasized <em>paragraph</em>.</p>

INCORRECT: overlapping elements

<p>here is an emphasized <em>paragraph.</p></em>

4.2. Element and attribute names must be in lower case

XHTML documents must use lower case for all HTML element and attribute names. This difference is necessary because XML is case-sensitive e.g. <li> and <LI> are different tags.

4.3. For non-empty elements, end tags are required

In SGML-based HTML 4 certain elements were permitted to omit the end tag; with the elements that followed implying closure. XML does not allow end tags to be omitted. All elements other than those declared in the DTD as `EMPTY` must have an end tag. Elements that are declared in the DTD as `EMPTY` can have an end tag *or* can use empty element shorthand (see Empty Elements).

CORRECT: terminated elements

<p>here is a paragraph.</p><p>here is another paragraph.</p>

INCORRECT: unterminated elements

<p>here is a paragraph.<p>here is another paragraph.

4.4. Attribute values must always be quoted

All attribute values must be quoted, even those which appear to be numeric.

CORRECT: quoted attribute values

<td rowspan="3">

INCORRECT: unquoted attribute values

<td rowspan=3>

4.5. Attribute Minimization

XML does not support attribute minimization. Attribute-value pairs must be written in full. Attribute names such as `compact` and `checked` cannot occur in elements without their value being specified.

CORRECT: unminimized attributes

<dl compact="compact">

INCORRECT: minimized attributes

<dl compact>

4.6. Empty Elements

Empty elements must either have an end tag or the start tag must end with `/>`. For instance, `<br/>` or `<hr></hr>`. See HTML Compatibility Guidelines for information on ways to ensure this is backward compatible with HTML 4 user agents.

CORRECT: terminated empty elements

<br/><hr/>

INCORRECT: unterminated empty elements

<br><hr>

4.7. White Space handling in attribute values

When user agents process attributes, they do so according to Section 3.3.3 of [XML]:

- Strip leading and trailing white space.
- Map sequences of one or more white space characters (including line breaks) to a single inter-word space.

4.8. Script and Style elements

In XHTML, the script and style elements are declared as having `#PCDATA` content. As a result, `<` and `&` will be treated as the start of markup, and entities such as `&lt;` and `&amp;` will be recognized as entity references by the XML processor to `<` and `&` respectively. Wrapping the content of the script or style element within a `CDATA` marked section avoids the expansion of these entities.

```
<script type="text/javascript">
<![CDATA[
... unescaped script content ...
]]>
</script>
```

`CDATA` sections are recognized by the XML processor and appear as nodes in the Document Object Model, see Section 1.3 of the DOM Level 1 Recommendation [DOM].

An alternative is to use external script and style documents.

4.9. SGML exclusions

SGML gives the writer of a DTD the ability to exclude specific elements from being contained within an element. Such prohibitions (called "exclusions") are not possible in XML.

For example, the HTML 4 Strict DTD forbids the nesting of an '`a`' element within another '`a`' element to any descendant depth. It is not possible to spell out such prohibitions in XML. Even though these prohibitions cannot be defined in the DTD, certain elements should not be nested. A summary of such elements and the elements that should not be nested in them is found in the normative Element Prohibitions.

4.10. The elements with 'id' and 'name' attributes

HTML 4 defined the `name` attribute for the elements `a`, `applet`, `form`, `frame`, `iframe`, `img`, and `map`. HTML 4 also introduced the `id` attribute. Both of these attributes are designed to be used as fragment identifiers.

In XML, fragment identifiers are of type `ID`, and there can only be a single attribute of type `ID` per element. Therefore, in XHTML 1.0 the `id` attribute is defined to be of type `ID`. In order to ensure that XHTML 1.0 documents are well-structured XML documents, XHTML 1.0 documents MUST use the `id` attribute when defining fragment identifiers on the elements listed above. See the HTML Compatibility Guidelines for information on ensuring such anchors are backward compatible when serving XHTML documents as media type `text/html`.

Note that in XHTML 1.0, the `name` attribute of these elements is formally deprecated, and will be removed in a subsequent version of XHTML.

4.11. Attributes with pre-defined value sets

HTML 4 and XHTML both have some attributes that have pre-defined and limited sets of values (e.g. the `type` attribute of the `input` element). In SGML and XML, these are called *enumerated attributes*. Under HTML 4, the interpretation of these values was *case-insensitive*, so a value of `TEXT` was equivalent to a value of `text`. Under XML, the interpretation of these values is *case-sensitive*, and in XHTML 1 all of these values are defined in lower-case.

4.12. Entity references as hex values

SGML and XML both permit references to characters by using hexadecimal values. In SGML these references could be made using either &#Xnn; or &#xnn;. In XML documents, you must use the lower-case version (i.e. &#xnn;)

5. Compatibility Issues

This section is normative.

Although there is no requirement for XHTML 1.0 documents to be compatible with existing user agents, in practice this is easy to accomplish. Guidelines for creating compatible documents can be found in Appendix C.

5.1. Internet Media Type

XHTML Documents which follow the guidelines set forth in Appendix C, "HTML Compatibility Guidelines" may be labeled with the Internet Media Type "text/html" [RFC2854], as they are compatible with most HTML browsers. Those documents, and any other document conforming to this specification, may also be labeled with the Internet Media Type "application/xhtml+xml" as defined in [RFC3236]. For further information on using media types with XHTML, see the informative note [XHTMLMIME].

A. DTDs

This appendix is normative.

These DTDs and entity sets form a normative part of this specification. The complete set of DTD files together with an XML declaration and SGML Open Catalog is included in the zip file and the gzip'd tar file for this specification. Users looking for local copies of the DTDs to work with should download and use those archives rather than using the specific DTDs referenced below.

A.1. Document Type Definitions

These DTDs approximate the HTML 4 DTDs. The W3C recommends that you use the authoritative versions of these DTDs at their defined SYSTEM identifiers when validating content. If you need to use these DTDs locally you should download one of the archives of this version. For completeness, the normative versions of the DTDs are included here:

A.1.1. XHTML-1.0-Strict

The file DTD/xhtml1-strict.dtd is a normative part of this specification. The annotated contents of this file are available in this separate section for completeness.

A.1.2. XHTML-1.0-Transitional

The file DTD/xhtml1-transitional.dtd is a normative part of this specification. The annotated contents of this file are available in this separate section for completeness.

A.1.3. XHTML-1.0-Frameset

The file DTD/xhtml1-frameset.dtd is a normative part of this specification. The annotated contents of this file are available in this separate section for completeness.

A.2. Entity Sets

The XHTML entity sets are the same as for HTML 4, but have been modified to be valid XML 1.0 entity declarations. Note the entity for the Euro currency sign (`&euro;` or `& #8364;` or `&#x20AC;`) is defined as part of the special characters.

A.2.1. Latin-1 characters

The file DTD/xhtml-lat1.ent is a normative part of this specification. The annotated contents of this file are available in this separate section for completeness.

A.2.2. Special characters

The file DTD/xhtml-special.ent is a normative part of this specification. The annotated contents of this file are available in this separate section for completeness.

A.2.3. Symbols

The file DTD/xhtml-symbol.ent is a normative part of this specification. The annotated contents of this file are available in this separate section for completeness.

B. Element Prohibitions

This appendix is normative.

The following elements have prohibitions on which elements they can contain (see SGML Exclusions). This prohibition applies to all depths of nesting, i.e. it contains all the descendant elements.

`a`
: must not contain other `a` elements.

`pre`
: must not contain the `img`, `object`, `big`, `small`, `sub`, or `sup` elements.

`button`
: must not contain the `input`, `select`, `textarea`, `label`, `button`, `form`, `fieldset`, `iframe` or `isindex` elements.

`label`
: must not contain other `label` elements.

`form`

must not contain other `form` elements.

C. HTML Compatibility Guidelines

This appendix is informative.

This appendix summarizes design guidelines for authors who wish their XHTML documents to render on existing HTML user agents. *Note that this recommendation does not define how HTML conforming user agents should process HTML documents. Nor does it define the meaning of the Internet Media Type `text/html`. For these definitions, see [HTML4] and [RFC2854] respectively.*

C.1. Processing Instructions and the XML Declaration

Be aware that processing instructions are rendered on some user agents. Also, some user agents interpret the XML declaration to mean that the document is unrecognized XML rather than HTML, and therefore may not render the document as expected. For compatibility with these types of legacy browsers, you may want to avoid using processing instructions and XML declarations. Remember, however, that when the XML declaration is not included in a document, the document can only use the default character encodings UTF-8 or UTF-16.

C.2. Empty Elements

Include a space before the trailing `/` and `>` of empty elements, e.g. `<br />`, `<hr />` and `<img src="karen.jpg" alt="Karen" />`. Also, use the minimized tag syntax for empty elements, e.g. `<br />`, as the alternative syntax `<br></br>` allowed by XML gives uncertain results in many existing user agents.

C.3. Element Minimization and Empty Element Content

Given an empty instance of an element whose content model is not `EMPTY` (for example, an empty title or paragraph) do not use the minimized form (e.g. use `< p> </p>` and not `<p />`).

C.4. Embedded Style Sheets and Scripts

Use external style sheets if your style sheet uses `<` or `&` or `]]>` or `--`. Use external scripts if your script uses `<` or `&` or `]]>` or `--`. Note that XML parsers are permitted to silently remove the contents of comments. Therefore, the historical practice of "hiding" scripts and style sheets within "comments" to make the documents backward compatible is likely to not work as expected in XML-based user agents.

C.5. Line Breaks within Attribute Values

Avoid line breaks and multiple white space characters within attribute values. These are handled inconsistently by user agents.

C.6. Isindex

Don't include more than one `isindex` element in the document `head`. The `isindex` element is deprecated in favor of the `input` element.

C.7. The `lang` and `xml:lang` Attributes

Use both the `lang` and `xml:lang` attributes when specifying the language of an element. The value of the `xml:lang` attribute takes precedence.

C.8. Fragment Identifiers

In XML, URI-references [RFC2396] that end with fragment identifiers of the form `"#foo"` do not refer to elements with an attribute `name="foo"`; rather, they refer to elements with an attribute defined to be of type `ID`, e.g., the `id` attribute in HTML 4. Many existing HTML clients don't support the use of `ID`-type attributes in this way, so identical values may be supplied for both of these attributes to ensure maximum forward and backward compatibility (e.g., `<a id="foo" name="foo">...</a>`).

Further, since the set of legal values for attributes of type `ID` is much smaller than for those of type `CDATA`, the type of the `name` attribute has been changed to `NMTOKEN`. This attribute is constrained such that it can only have the same values as type `ID`, or as the `Name` production in XML 1.0 Section 2.3, production 5. Unfortunately, this constraint cannot be expressed in the XHTML 1.0 DTDs. Because of this change, care must be taken when converting existing HTML documents. The values of these attributes must be unique within the document, valid, and any references to these fragment identifiers (both internal and external) must be updated should the values be changed during conversion.

Note that the collection of legal values in XML 1.0 Section 2.3, production 5 is much larger than that permitted to be used in the `ID` and `NAME` types defined in HTML 4. When defining fragment identifiers to be backward-compatible, only strings matching the pattern `[A-Za-z][A-Za-z0-9:_.-]*` should be used. See Section 6.2 of [HTML4] for more information.

Finally, note that XHTML 1.0 has deprecated the `name` attribute of the `a`, `applet`, `form`, `frame`, `iframe`, `img`, and `map` elements, and it will be removed from XHTML in subsequent versions.

C.9. Character Encoding

Historically, the character encoding of an HTML document is either specified by a web server via the charset parameter of the HTTP Content-Type header, or via a `meta` element in the document itself. In an XML document, the character encoding of the document is specified on the XML declaration (e.g., `<?xml version="1.0" encoding="EUC-JP"?>`). In order to portably present documents with specific character encodings, the best approach is to ensure that the web server provides the correct headers. If this is not possible, a document that wants to set its character encoding explicitly must include both the XML declaration an encoding declaration and a `meta` http-equiv statement (e.g., `<meta http-equiv="Content-type" content="text/html; charset=EUC-JP" />`). In XHTML-conforming user agents, the value of the encoding declaration of the XML declaration takes precedence.

Note: be aware that if a document must include the character encoding declaration in a meta http-equiv statement, that document may always be interpreted by HTTP servers and/or user agents as being of the internet media type defined in that statement. If a document is to be served as multiple media types, the HTTP server must be used to set the encoding of the document.

C.10. Boolean Attributes

Some HTML user agents are unable to interpret boolean attributes when these appear in their full (non-minimized) form, as required by XML 1.0. Note this problem doesn't affect user agents compliant with HTML 4. The following attributes are involved: `compact`, `nowrap`, `ismap`, `declare`, `noshade`, `checked`, `disabled`, `readonly`, `multiple`, `selected`, `noresize`, `defer`.

C.11. Document Object Model and XHTML

The Document Object Model level 1 Recommendation [DOM] defines document object model interfaces for XML and HTML 4. The HTML 4 document object model specifies that HTML element and attribute names are returned in upper-case. The XML document object model specifies that element and attribute names are returned in the case they are specified. In XHTML 1.0, elements and attributes are specified in lower-case. This apparent difference can be addressed in two ways:

1. User agents that access XHTML documents served as Internet media type `text/html` via the DOM can use the HTML DOM, and can rely upon element and attribute names being returned in upper-case from those interfaces.
2. User agents that access XHTML documents served as Internet media types `text/xml`, `application/xml`, or `application/xhtml+xml` can also use the XML DOM. Elements and attributes will be returned in lower-case. Also, some XHTML elements may or may not appear in the object tree because they are optional in the content model (e.g. the `tbody` element within `table`). This occurs because in HTML 4 some elements were permitted to be minimized such that their start and end tags are both omitted (an SGML feature). This is not possible in XML. Rather than require document authors to insert extraneous elements, XHTML has made the elements optional. User agents need to adapt to this accordingly. For further information on this topic, see [DOM2]

C.12. Using Ampersands in Attribute Values (and Elsewhere)

In both SGML and XML, the ampersand character ("&") declares the beginning of an entity reference (e.g., &reg; for the registered trademark symbol "®"). Unfortunately, many HTML user agents have silently ignored incorrect usage of the ampersand character in HTML documents - treating ampersands that do not look like entity references as literal ampersands. XML-based user agents will not tolerate this incorrect usage, and any document that uses an ampersand incorrectly will not be "valid", and consequently will not conform to this specification. In order to ensure that documents are compatible with historical HTML user agents and XML-based user agents, ampersands used in a document that are to be treated as literal characters must be expressed themselves as an entity reference (e.g. "`&amp;`"). For example, when the `href` attribute of the `a` element refers to a CGI script that takes parameters, it must be expressed as `http://my.site.dom/cgi-bin/myscript.pl?class=guest&amp;name=user` rather than as `http://my.site.dom/cgi-bin/myscript.pl?class=guest&name=user`.

C.13. Cascading Style Sheets (CSS) and XHTML

The Cascading Style Sheets level 2 Recommendation [CSS2] defines style properties which are applied to the parse tree of the HTML or XML documents. Differences in parsing will produce different visual or aural results, depending on the selectors used. The following hints will reduce this effect for documents which are served without modification as both media types:

1. CSS style sheets for XHTML should use lower case element and attribute names.
2. In tables, the tbody element will be inferred by the parser of an HTML user agent, but not by the parser of an XML user agent. Therefore you should always explicitly add a tbody element if it is referred to in a CSS selector.
3. Within the XHTML namespace, user agents are expected to recognize the "id" attribute as an attribute of type ID. Therefore, style sheets should be able to continue using the shorthand "#" selector syntax even if the user agent does not read the DTD.
4. Within the XHTML namespace, user agents are expected to recognize the "class" attribute. Therefore, style sheets should be able to continue using the shorthand "." selector syntax.
5. CSS defines different conformance rules for HTML and XML documents; be aware that the HTML rules apply to XHTML documents delivered as HTML and the XML rules apply to XHTML documents delivered as XML.

C.14. Referencing Style Elements when serving as XML

In HTML 4 and XHTML, the `style` element can be used to define document-internal style rules. In XML, an XML stylesheet declaration is used to define style rules. In order to be compatible with this convention, `style` elements should have their fragment identifier set using the `id` attribute, and an XML stylesheet declaration should reference this fragment. For example:

```
<?xml-stylesheet href="http://www.w3.org/StyleSheets/TR/W3C-REC.css" type="text/css"?>
<?xml-stylesheet href="#internalStyle" type="text/css"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>An internal stylesheet example</title>
<style type="text/css" id="internalStyle">
  code {
    color: green;
    font-family: monospace;
    font-weight: bold;
  }
</style>
</head>
<body>
<p>
  This is text that uses our
  <code>internal stylesheet</code>.
</p>
</body>
</html>
```

C.15. White Space Characters in HTML vs. XML

Some characters that are legal in HTML documents, are illegal in XML document. For example, in HTML, the Formfeed character (U+000C) is treated as white space, in XHTML, due to XML's definition of characters, it is illegal.

C.16. The Named Character Reference &apos;

The named character reference `&apos;` (the apostrophe, U+0027) was introduced in XML 1.0 but does not appear in HTML. Authors should therefore use `&#39;` instead of `&apos;` to work as expected in HTML 4 user agents.

D. Acknowledgements

This appendix is informative.

This specification was written with the participation of the members of the W3C HTML Working Group.

At publication of the second edition, the membership was:

Steven Pemberton, CWI/W3C (HTML Working Group Chair)
Daniel Austin, Grainger
Jonny Axelsson, Opera Software
Tantek Çelik, Microsoft
Doug Dominiak, Openwave Systems
Herman Elenbaas, Philips Electronics
Beth Epperson, Netscape/AOL
Masayasu Ishikawa, W3C (HTML Activity Lead)
Shin'ichi Matsui, Panasonic
Shane McCarron, Applied Testing and Technology
Ann Navarro, WebGeek, Inc.
Subramanian Peruvemba, Oracle
Rob Relyea, Microsoft
Sebastian Schnitzenbaumer, SAP
Peter Stark, Sony Ericsson

At publication of the first edition, the membership was:

Steven Pemberton, CWI (HTML Working Group Chair)
Murray Altheim, Sun Microsystems
Daniel Austin, AskJeeves (CNET: The Computer Network through July 1999)
Frank Boumphrey, HTML Writers Guild
John Burger, Mitre
Andrew W. Donoho, IBM
Sam Dooley, IBM
Klaus Hofrichter, GMD
Philipp Hoschka, W3C
Masayasu Ishikawa, W3C
Warner ten Kate, Philips Electronics

Peter King, Phone.com
Paula Klante, JetForm
Shin'ichi Matsui, Panasonic (W3C visiting engineer through September 1999)
Shane McCarron, Applied Testing and Technology (The Open Group through August 1999)
Ann Navarro, HTML Writers Guild
Zach Nies, Quark
Dave Raggett, W3C/HP (HTML Activity Lead)
Patrick Schmitz, Microsoft
Sebastian Schnitzenbaumer, Stack Overflow
Peter Stark, Phone.com
Chris Wilson, Microsoft
Ted Wugofski, Gateway 2000
Dan Zigmond, WebTV Networks

E. References

This appendix is informative.

[CSS2]
"*Cascading Style Sheets, level 2 (CSS2) Specification*", B. Bos, H. W. Lie, C. Lilley, I. Jacobs, 12 May 1998.
Latest version available at: http://www.w3.org/TR/REC-CSS2

[DOM]
"*Document Object Model (DOM) Level 1 Specification*", Lauren Wood *et al.*, 1 October 1998.
Latest version available at: http://www.w3.org/TR/REC-DOM-Level-1

[DOM2]
"*Document Object Model (DOM) Level 2 Core Specification*", A. Le Hors, *et al.*, 13 November 2000.
Latest version available at: http://www.w3.org/TR/DOM-Level-2-Core

[HTML]
"*HTML 4.01 Specification*", D. Raggett, A. Le Hors, I. Jacobs, 24 December 1999.
Latest version available at: http://www.w3.org/TR/html401

[POSIX.1]
"*ISO/IEC 9945-1:1990 Information Technology - Portable Operating System Interface (POSIX) - Part 1: System Application Program Interface (API) [C Language]*", Institute of Electrical and Electronics Engineers, Inc, 1990.

[RFC2045]
"*Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*", N. Freed and N. Borenstein, November 1996. Note that this RFC obsoletes RFC1521, RFC1522, and RFC1590.

[RFC2046]
"*RFC2046: Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*", N. Freed and N. Borenstein, November 1996.
Available at http://www.ietf.org/rfc/rfc2046.txt. Note that this RFC obsoletes RFC1521, RFC1522, and RFC1590.

[RFC2119]
"*RFC2119: Key words for use in RFCs to Indicate Requirement Levels*", S. Bradner, March 1997.
Available at: http://www.ietf.org/rfc/rfc2119.txt

[RFC2376]

"*RFC2376: XML Media Types*", E. Whitehead, M. Murata, July 1998.
This document is obsoleted by [RFC3023].
Available at: http://www.ietf.org/rfc/rfc2376.txt

[RFC2396]
"*RFC2396: Uniform Resource Identifiers (URI): Generic Syntax*", T. Berners-Lee, R. Fielding, L. Masinter, August 1998.
This document updates RFC1738 and RFC1808.
Available at: http://www.ietf.org/rfc/rfc2396.txt

[RFC2854]
"*RFC2854: The text/html Media Type*", D. Conolly, L. Masinter, June 2000.
Available at: http://www.ietf.org/rfc/rfc2854.txt

[RFC3023]
"*RFC3023: XML Media Types*", M. Murata, S. St.Laurent, D. Kohn, January 2001.
This document obsoletes [RFC2376].
Available at: http://www.ietf.org/rfc/rfc3023.txt

[RFC3066]
"Tags for the Identification of Languages", H. Alvestrand, January 2001.
Available at: http://www.ietf.org/rfc/rfc3066.txt

[RFC3236]
"The 'application/xhtml+xml' Media Type", M. Baker, P. Stark, January 2002.
Available at: http://www.ietf.org/rfc/rfc3236.txt

[XHTML+MathML]
"XHTML plus Math 1.1 DTD", "A.2 MathML as a DTD Module", Mathematical Markup Language (MathML) Version 2.0. Available at: http://www.w3.org/TR/MathML2/dtd/xhtml-math11-f.dtd

[XHTMLMIME]
"*XHTML Media Types*", Masayasu Ishikawa, 1 August 2002.
Latest version available at: http://www.w3.org/TR/xhtml-media-types

[XHTMLMOD]
"*Modularization of XHTML*", M. Altheim et al., 10 April 2001.
Latest version available at: http://www.w3.org/TR/xhtml-modularization

[XML]
"Extensible Markup Language (XML) 1.0 Specification (Second Edition)", T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, 6 October 2000.
Latest version available at: http://www.w3.org/TR/REC-xml

[XMLNS]
"Namespaces in XML", T. Bray, D. Hollander, A. Layman, 14 January 1999.
XML namespaces provide a simple method for qualifying names used in XML documents by associating them with namespaces identified by URI.
Latest version available at: http://www.w3.org/TR/REC-xml-names

[XMLC14N]
"Canonical XML Version 1.0", J. Boyer, 15 March 2001.
This document describes a method for generating a physical representation, the canonical form, of an XML document.
Latest version available at: http://www.w3.org/TR/xml-c14n

W3C WAI-AAA WCAG 1.0

Simple Object Access Protocol (SOAP) 1.1

W3C Note 08 May 2000

This version:
http://www.w3.org/TR/2000/NOTE-SOAP-20000508
Latest version:
http://www.w3.org/TR/SOAP

Authors (alphabetically):
Don Box, DevelopMentor
David Ehnebuske, IBM
Gopal Kakivaya, Microsoft
Andrew Layman, Microsoft
Noah Mendelsohn, Lotus Development Corp.
Henrik Frystyk Nielsen, Microsoft
Satish Thatte, Microsoft
Dave Winer, UserLand Software, Inc.

Abstract

SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses. SOAP can potentially be used in combination with a variety of other protocols; however, the only bindings defined in this document describe how to use SOAP in combination with HTTP and HTTP Extension Framework.

Status

This document is a submission to the World Wide Web Consortium (see Submission Request, W3C Staff Comment) to propose the formation of a working group in the area of XML-based protocols. Comments are welcome to the authors but you are encouraged to share your views on the W3C's public mailing list <xml-dist-app@w3.org> (see archives).

This document is a NOTE made available by the W3C for discussion only. Publication of this Note by W3C indicates no endorsement by W3C or the W3C Team, or any W3C Members. W3C has had no editorial control over the preparation of this Note. This document is a work in progress and may be updated, replaced, or rendered obsolete by other documents at any time.

A list of current W3C technical documents can be found at the Technical Reports page.

Table of Contents

1. Introduction

SOAP provides a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralized, distributed environment using XML. SOAP does not itself define any application semantics such as a programming model or implementation specific semantics; rather it defines a simple mechanism for expressing application semantics by providing a modular packaging model and encoding mechanisms for encoding data within modules. This allows SOAP to be used in a large variety of systems ranging from messaging systems to RPC.

SOAP consists of three parts:

- The SOAP envelope (see section 4) construct defines an overall framework for expressing **what** is in a message; **who** should deal with it, and **whether** it is optional or mandatory.
- The SOAP encoding rules (see section 5) defines a serialization mechanism that can be used to exchange instances of application-defined datatypes.
- The SOAP RPC representation (see section 7) defines a convention that can be used to represent remote procedure calls and responses.

Although these parts are described together as part of SOAP, they are functionally orthogonal. In particular, the envelope and the encoding rules are defined in different namespaces in order to promote simplicity through modularity.

In addition to the SOAP envelope, the SOAP encoding rules and the SOAP RPC conventions, this specification defines two protocol bindings that describe how a SOAP message can be carried in HTTP [5] messages either with or without the HTTP Extension Framework [6].

1.1 Design Goals

A major design goal for SOAP is simplicity and extensibility. This means that there are several features from traditional messaging systems and distributed object systems that are not part of the core SOAP specification. Such features include

- Distributed garbage collection
- Boxcarring or batching of messages
- Objects-by-reference (which requires distributed garbage collection)
- Activation (which requires objects-by-reference)

1.2 Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [2].

The namespace prefixes "SOAP-ENV" and "SOAP-ENC" used in this document are associated with the SOAP namespaces "http://schemas.xmlsoap.org/soap/envelope/" and "http://schemas.xmlsoap.org/soap/encoding/" respectively.

Throughout this document, the namespace prefix "xsi" is assumed to be associated with the URI "http://www.w3.org/1999/XMLSchema-instance" which is defined in the XML Schemas specification [11]. Similarly, the namespace prefix "xsd" is assumed to be associated with the URI "http://www.w3.org/1999/XMLSchema" which is defined in [10]. The namespace prefix "tns" is used to indicate whatever is the target namespace of the current document. All other namespace prefixes are samples only.

Namespace URIs of the general form "some-URI" represent some application-dependent or context-dependent URI [4].

This specification uses the augmented Backus-Naur Form (BNF) as described in RFC-2616 [5] for certain constructs.

1.3 Examples of SOAP Messages

In this example, a GetLastTradePrice SOAP request is sent to a StockQuote service. The request takes a string parameter, ticker symbol, and returns a float in the SOAP response. The SOAP Envelope element is the top element of the XML document representing the SOAP message. XML namespaces are used to disambiguate SOAP identifiers from application specific identifiers. The example illustrates the HTTP bindings defined in section 6. It is worth noting that the rules governing XML payload format in SOAP are entirely independent of the fact that the payload is carried in HTTP.

More examples are available in Appendix A.

Example 1 SOAP Message Embedded in HTTP Request

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
       <m:GetLastTradePrice xmlns:m="Some-URI">
           <symbol>DIS</symbol>
```

```
        </m:GetLastTradePrice>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Following is the response message containing the HTTP message with the SOAP message as the payload:

Example 2 SOAP Message Embedded in HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   <SOAP-ENV:Body>
       <m:GetLastTradePriceResponse xmlns:m="Some-URI">
           <Price>34.5</Price>
       </m:GetLastTradePriceResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

2. The SOAP Message Exchange Model

SOAP messages are fundamentally one-way transmissions from a sender to a receiver, but as illustrated above, SOAP messages are often combined to implement patterns such as request/response.

SOAP implementations can be optimized to exploit the unique characteristics of particular network systems. For example, the HTTP binding described in section 6 provides for SOAP response messages to be delivered as HTTP responses, using the same connection as the inbound request.

Regardless of the protocol to which SOAP is bound, messages are routed along a so-called "message path", which allows for processing at one or more intermediate nodes in addition to the ultimate destination.

A SOAP application receiving a SOAP message MUST process that message by performing the following actions in the order listed below:

1. Identify all parts of the SOAP message intended for that application (see section 4.2.2)
2. Verify that all mandatory parts identified in step 1 are supported by the application for this message (see section 4.2.3) and process them accordingly. If this is not the case then discard the message (see section 4.4). The processor MAY ignore optional parts identified in step 1 without affecting the outcome of the processing.
3. If the SOAP application is not the ultimate destination of the message then remove all parts identified in step 1 before forwarding the message.

Processing a message or a part of a message requires that the SOAP processor understands, among other things, the exchange pattern being used (one way, request/response, multicast, etc.), the role of the recipient in that pattern, the employment (if any) of RPC mechanisms such as the one documented in section 7, the representation or encoding of data, as well as other semantics necessary for correct processing.

While attributes such as the SOAP encodingStyle attribute (see section 4.1.1) can be used to describe certain aspects of a message, this specification does not mandate a particular means by which the recipient makes such determinations in general. For example, certain applications will understand that a particular <getStockPrice> element signals an RPC request using the conventions of section 7, while another application may infer that all traffic directed to it is encoded as one way messages.

3. Relation to XML

All SOAP messages are encoded using XML (see [7] for more information on XML).

A SOAP application SHOULD include the proper SOAP namespace on all elements and attributes defined by SOAP in messages that it generates. A SOAP application MUST be able to process SOAP namespaces in messages that it receives. It MUST discard messages that have incorrect namespaces (see section 4.4) and it MAY process SOAP messages without SOAP namespaces as though they had the correct SOAP namespaces.

SOAP defines two namespaces (see [8] for more information on XML namespaces):

- The SOAP envelope has the namespace identifier "http://schemas.xmlsoap.org/soap/envelope/"
- The SOAP serialization has the namespace identifier "http://schemas.xmlsoap.org/soap/encoding/"

A SOAP message MUST NOT contain a Document Type Declaration. A SOAP message MUST NOT contain Processing Instructions. [7]

SOAP uses the local, unqualified "id" attribute of type "ID" to specify the unique identifier of an encoded element. SOAP uses the local, unqualified attribute "href" of type "uri-reference" to specify a reference to that value, in a manner conforming to the XML Specification [7], XML Schema Specification [11], and XML Linking Language Specification [9].

With the exception of the SOAP mustUnderstand attribute (see section 4.2.3) and the SOAP actor attribute (see section 4.2.2), it is generally permissible to have attributes and their values appear in XML instances or alternatively in schemas, with equal effect. That is, declaration in a DTD or schema with a default or fixed value is semantically equivalent to appearance in an instance.

4. SOAP Envelope

A SOAP message is an XML document that consists of a mandatory SOAP envelope, an optional SOAP header, and a mandatory SOAP body. This XML document is referred to as a SOAP message for the rest of this specification. The namespace identifier for the elements and attributes defined in this section is "http://schemas.xmlsoap.org/soap/envelope/". A SOAP message contains the following:

- The Envelope is the top element of the XML document representing the message.
- The Header is a generic mechanism for adding features to a SOAP message in a decentralized manner without prior agreement between the communicating parties. SOAP defines a few attributes that can be used to indicate who should deal with a feature and whether it is optional or mandatory (see section 4.2)
- The Body is a container for mandatory information intended for the ultimate recipient of the message (see section 4.3). SOAP defines one element for the body, which is the Fault element used for reporting errors.

The grammar rules are as follows:

1. Envelope
   - The element name is "Envelope".
   - The element MUST be present in a SOAP message
   - The element MAY contain namespace declarations as well as additional attributes. If present, such additional attributes MUST be namespace-qualified. Similarly, the element MAY contain additional sub elements. If present these elements MUST be namespace-qualified and MUST follow the SOAP Body element.
2. Header (see section 4.2)
   - The element name is "Header".
   - The element MAY be present in a SOAP message. If present, the element MUST be the first immediate child element of a SOAP Envelope element.
   - The element MAY contain a set of header entries each being an immediate child element of the SOAP Header element. All immediate child elements of the SOAP Header element MUST be namespace-qualified.
3. Body (see section 4.3)
   - The element name is "Body".
   - The element MUST be present in a SOAP message and MUST be an immediate child element of a SOAP Envelope element. It MUST directly follow the SOAP Header element if present. Otherwise it MUST be the first immediate child element of the SOAP Envelope element.
   - The element MAY contain a set of body entries each being an immediate child element of the SOAP Body element. Immediate child elements of the SOAP Body element MAY be namespace-qualified. SOAP defines the SOAP Fault element, which is used to indicate error messages (see section 4.4).

4.1.1 SOAP encodingStyle Attribute

The SOAP encodingStyle global attribute can be used to indicate the serialization rules used in a SOAP message. This attribute MAY appear on any element, and is scoped to that element's contents and all child elements not themselves containing such an attribute, much as an XML namespace declaration is scoped. There is no default encoding defined for a SOAP message.

The attribute value is an ordered list of one or more URIs identifying the serialization rule or rules that can be used to deserialize the SOAP message indicated in the order of most specific to least specific. Examples of values are

```
"http://schemas.xmlsoap.org/soap/encoding/"
"http://my.host/encoding/restricted http://my.host/encoding/"
""
```

The serialization rules defined by SOAP in section 5 are identified by the URI "http://schemas.xmlsoap.org/soap/encoding/". Messages using this particular serialization SHOULD indicate this using the SOAP encodingStyle attribute. In addition, all URIs syntactically beginning with "http://schemas.xmlsoap.org/soap/encoding/" indicate conformance with the SOAP encoding rules defined in section 5 (though with potentially tighter rules added).

A value of the zero-length URI ("") explicitly indicates that no claims are made for the encoding style of contained elements. This can be used to turn off any claims from containing elements.

4.1.2 Envelope Versioning Model

SOAP does not define a traditional versioning model based on major and minor version numbers. A SOAP message MUST have an Envelope element associated with the "http://schemas.xmlsoap.org/soap/envelope/" namespace. If a message is received by a SOAP application in which the SOAP Envelope element is associated with a different namespace, the application MUST treat this as a version error and discard the message. If the message is received through a request/response protocol such as HTTP, the application MUST respond with a SOAP VersionMismatch faultcode message (see section 4.4) using the SOAP "http://schemas.xmlsoap.org/soap/envelope/" namespace.

4.2 SOAP Header

SOAP provides a flexible mechanism for extending a message in a decentralized and modular way without prior knowledge between the communicating parties. Typical examples of extensions that can be implemented as header entries are authentication, transaction management, payment etc.

The Header element is encoded as the first immediate child element of the SOAP Envelope XML element. All immediate child elements of the Header element are called header entries.

The encoding rules for header entries are as follows:

1. A header entry is identified by its fully qualified element name, which consists of the namespace URI and the local name. All immediate child elements of the SOAP Header element MUST be namespace-qualified.
2. The SOAP encodingStyle attribute MAY be used to indicate the encoding style used for the header entries (see section 4.1.1).

3. The SOAP mustUnderstand attribute (see section 4.2.3) and SOAP actor attribute (see section 4.2.2) MAY be used to indicate how to process the entry and by whom (see section 4.2.1).

4.2.1 Use of Header Attributes

The SOAP Header attributes defined in this section determine how a recipient of a SOAP message should process the message as described in section 2. A SOAP application generating a SOAP message SHOULD only use the SOAP Header attributes on immediate child elements of the SOAP Header element. The recipient of a SOAP message MUST ignore all SOAP Header attributes that are not applied to an immediate child element of the SOAP Header element.

An example is a header with an element identifier of "Transaction", a "mustUnderstand" value of "1", and a value of 5. This would be encoded as follows:

```
<SOAP-ENV:Header>
   <t:Transaction
      xmlns:t="some-URI" SOAP-ENV:mustUnderstand="1">
          5
   </t:Transaction>
</SOAP-ENV:Header>
```

4.2.2 SOAP actor Attribute

A SOAP message travels from the originator to the ultimate destination, potentially by passing through a set of SOAP intermediaries along the message path. A SOAP intermediary is an application that is capable of both receiving and forwarding SOAP messages. Both intermediaries as well as the ultimate destination are identified by a URI.

Not all parts of a SOAP message may be intended for the ultimate destination of the SOAP message but, instead, may be intended for one or more of the intermediaries on the message path. The role of a recipient of a header element is similar to that of accepting a contract in that it cannot be extended beyond the recipient. That is, a recipient receiving a header element MUST NOT forward that header element to the next application in the SOAP message path. The recipient MAY insert a similar header element but in that case, the contract is between that application and the recipient of that header element.

The SOAP actor global attribute can be used to indicate the recipient of a header element. The value of the SOAP actor attribute is a URI. The special URI "http://schemas.xmlsoap.org/soap/actor/next" indicates that the header element is intended for the very first SOAP application that processes the message. This is similar to the hop-by-hop scope model represented by the Connection header field in HTTP.

Omitting the SOAP actor attribute indicates that the recipient is the ultimate destination of the SOAP message.

This attribute MUST appear in the SOAP message instance in order to be effective (see section 3 and 4.2.1).

4.2.3 SOAP mustUnderstand Attribute

The SOAP mustUnderstand global attribute can be used to indicate whether a header entry is mandatory or optional for the recipient to process. The recipient of a header entry is defined by the SOAP actor attribute (see section 4.2.2). The value of the mustUnderstand attribute is either "1" or "0". The absence of the SOAP mustUnderstand attribute is semantically equivalent to its presence with the value "0".

If a header element is tagged with a SOAP mustUnderstand attribute with a value of "1", the recipient of that header entry either MUST obey the semantics (as conveyed by the fully qualified name of the element) and process correctly to those semantics, or MUST fail processing the message (see section 4.4).

The SOAP mustUnderstand attribute allows for robust evolution. Elements tagged with the SOAP mustUnderstand attribute with a value of "1" MUST be presumed to somehow modify the semantics of their parent or peer elements. Tagging elements in this manner assures that this change in semantics will not be silently (and, presumably, erroneously) ignored by those who may not fully understand it.

This attribute MUST appear in the instance in order to be effective (see section 3 and 4.2.1).

4.3 SOAP Body

The SOAP Body element provides a simple mechanism for exchanging mandatory information intended for the ultimate recipient of the message. Typical uses of the Body element include marshalling RPC calls and error reporting.

The Body element is encoded as an immediate child element of the SOAP Envelope XML element. If a Header element is present then the Body element MUST immediately follow the Header element, otherwise it MUST be the first immediate child element of the Envelope element.

All immediate child elements of the Body element are called body entries and each body entry is encoded as an independent element within the SOAP Body element.

The encoding rules for body entries are as follows:

1. A body entry is identified by its fully qualified element name, which consists of the namespace URI and the local name. Immediate child elements of the SOAP Body element MAY be namespace-qualified.
2. The SOAP encodingStyle attribute MAY be used to indicate the encoding style used for the body entries (see section 4.1.1).

SOAP defines one body entry, which is the Fault entry used for reporting errors (see section 4.4).

4.3.1 Relationship between SOAP Header and Body

While the Header and Body are defined as independent elements, they are in fact related. The relationship between a body entry and a header entry is as follows: A body entry is semantically equivalent to a header entry intended for the default actor and with a SOAP mustUnderstand attribute with a value of "1". The default actor is indicated by not using the actor attribute (see section 4.2.2).

4.4 SOAP Fault

The SOAP Fault element is used to carry error and/or status information within a SOAP message. If present, the SOAP Fault element MUST appear as a body entry and MUST NOT appear more than once within a Body element.

The SOAP Fault element defines the following four subelements:

faultcode
: The faultcode element is intended for use by software to provide an algorithmic mechanism for identifying the fault. The faultcode MUST be present in a SOAP Fault element and the faultcode value MUST be a qualified name as defined in [8], section 3. SOAP defines a small set of SOAP fault codes covering basic SOAP faults (see section 4.4.1)

faultstring
: The faultstring element is intended to provide a human readable explanation of the fault and is not intended for algorithmic processing. The faultstring element is similar to the 'Reason-Phrase' defined by HTTP (see [5], section 6.1). It MUST be present in a SOAP Fault element and SHOULD provide at least some information explaining the nature of the fault.

faultactor
: The faultactor element is intended to provide information about who caused the fault to happen within the message path (see section 2). It is similar to the SOAP actor attribute (see section 4.2.2) but instead of indicating the destination of the header entry, it indicates the source of the fault. The value of the faultactor attribute is a URI identifying the source. Applications that do not act as the ultimate destination of the SOAP message MUST include the faultactor element in a SOAP Fault element. The ultimate destination of a message MAY use the faultactor element to indicate explicitly that it generated the fault (see also the detail element below).

detail
: The detail element is intended for carrying application specific error information related to the Body element. It MUST be present if the contents of the Body element could not be successfully processed. It MUST NOT be used to carry information about error information belonging to header entries. Detailed error information belonging to header entries MUST be carried within header entries.

The absence of the detail element in the Fault element indicates that the fault is not related to processing of the Body element. This can be used to distinguish whether the Body element was processed or not in case of a fault situation.

All immediate child elements of the detail element are called detail entries and each detail entry is encoded as an independent element within the detail element.

The encoding rules for detail entries are as follows (see also example 10):

1. A detail entry is identified by its fully qualified element name, which consists of the namespace URI and the local name. Immediate child elements of the detail element MAY be namespace-qualified.

2. The SOAP encodingStyle attribute MAY be used to indicate the encoding style used for the detail entries (see section 4.1.1).

Other Fault subelements MAY be present, provided they are namespace-qualified.

4.4.1 SOAP Fault Codes

The faultcode values defined in this section MUST be used in the faultcode element when describing faults defined by this specification. The namespace identifier for these faultcode values is "http://schemas.xmlsoap.org/soap/envelope/". Use of this space is recommended (but not required) in the specification of methods defined outside of the present specification.

The default SOAP faultcode values are defined in an extensible manner that allows for new SOAP faultcode values to be defined while maintaining backwards compatibility with existing faultcode values. The mechanism used is very similar to the 1xx, 2xx, 3xx etc basic status classes classes defined in HTTP (see [5] section 10). However, instead of integers, they are defined as XML qualified names (see [8] section 3). The character "." (dot) is used as a separator of faultcode values indicating that what is to the left of the dot is a more generic fault code value than the value to the right. Example

```
Client.Authentication
```

The set of faultcode values defined in this document is:

Name	Meaning
VersionMismatch	The processing party found an invalid namespace for the SOAP Envelope element (see section 4.1.2)
MustUnderstand	An immediate child element of the SOAP Header element that was either not understood or not obeyed by the processing party contained a SOAP mustUnderstand attribute with a value of "1" (see section 4.2.3)
Client	The Client class of errors indicate that the message was incorrectly formed or did not contain the appropriate information in order to succeed. For example, the message could lack the proper authentication or payment information. It is generally an indication that the message should not be resent without change. See also section 4.4 for a description of the SOAP Fault detail sub-element.
Server	The Server class of errors indicate that the message could not be processed for reasons not directly attributable to the contents of the message itself but rather to the processing of the message. For example, processing could include communicating with an upstream processor, which didn't respond. The message may succeed at a later point in time. See also section 4.4 for a description of the SOAP Fault detail sub-element.

5. SOAP Encoding

The SOAP encoding style is based on a simple type system that is a generalization of the common features found in type systems in programming languages, databases and semi-structured data. A type either is a simple (scalar) type or is a compound type constructed as a composite of several parts, each with a type. This is described in more detail below. This section defines rules for serialization of a graph of typed objects. It operates on two levels. First, given a schema in any notation consistent with the type system described, a schema for an XML grammar may be constructed. Second, given a type-system schema and a particular graph of values conforming to that schema, an XML instance may be constructed. In reverse, given an XML instance produced in accordance with these rules, and given also the original schema, a copy of the original value graph may be constructed.

The namespace identifier for the elements and attributes defined in this section is "http://schemas.xmlsoap.org/soap/encoding/". The encoding samples shown assume all namespace declarations are at a higher element level.

Use of the data model and encoding style described in this section is encouraged but not required; other data models and encodings can be used in conjunction with SOAP (see section 4.1.1).

5.1 Rules for Encoding Types in XML

XML allows very flexible encoding of data. SOAP defines a narrower set of rules for encoding. This section defines the encoding rules at a high level, and the next section describes the encoding rules for specific types when they require more detail. The encodings described in this section can be used in conjunction with the mapping of RPC calls and responses specified in Section 7.

To describe encoding, the following terminology is used:

1. A "value" is a string, the name of a measurement (number, date, enumeration, etc.) or a composite of several such primitive values. All values are of specific types.
2. A "simple value" is one without named parts. Examples of simple values are particular strings, integers, enumerated values etc.
3. A "compound value" is an aggregate of relations to other values. Examples of Compound Values are particular purchase orders, stock reports, street addresses, etc.
4. Within a compound value, each related value is potentially distinguished by a role name, ordinal or both. This is called its "accessor." Examples of compound values include particular Purchase Orders, Stock Reports etc. Arrays are also compound values. It is possible to have compound values with several accessors each named the same, as for example, RDF does.
5. An "array" is a compound value in which ordinal position serves as the only distinction among member values.
6. A "struct" is a compound value in which accessor name is the only distinction among member values, and no accessor has the same name as any other.
7. A "simple type" is a class of simple values. Examples of simple types are the classes called "string," "integer," enumeration classes, etc.

8. A "compound type" is a class of compound values. An example of a compound type is the class of purchase order values sharing the same accessors (shipTo, totalCost, etc.) though with potentially different values (and perhaps further constrained by limits on certain values).
9. Within a compound type, if an accessor has a name that is distinct within that type but is not distinct with respect to other types, that is, the name plus the type together are needed to make a unique identification, the name is called "locally scoped." If however the name is based in part on a Uniform Resource Identifier, directly or indirectly, such that the name alone is sufficient to uniquely identify the accessor irrespective of the type within which it appears, the name is called "universally scoped."
10. Given the information in the schema relative to which a graph of values is serialized, it is possible to determine that some values can only be related by a single instance of an accessor. For others, it is not possible to make this determination. If only one accessor can reference it, a value is considered "single-reference". If referenced by more than one, actually or potentially, it is "multi-reference." Note that it is possible for a certain value to be considered "single-reference" relative to one schema and "multi-reference" relative to another.
11. Syntactically, an element may be "independent" or "embedded." An independent element is any element appearing at the top level of a serialization. All others are embedded elements.

Although it is possible to use the xsi:type attribute such that a graph of values is self-describing both in its structure and the types of its values, the serialization rules permit that the types of values MAY be determinate only by reference to a schema. Such schemas MAY be in the notation described by "XML Schema Part 1: Structures" [10] and "XML Schema Part 2: Datatypes" [11] or MAY be in any other notation. Note also that, while the serialization rules apply to compound types other than arrays and structs, many schemas will contain only struct and array types.

The rules for serialization are as follows:

1. All values are represented as element content. A multi-reference value MUST be represented as the content of an independent element. A single-reference value SHOULD not be (but MAY be).
2. For each element containing a value, the type of the value MUST be represented by at least one of the following conditions: (a) the containing element instance contains an xsi:type attribute, (b) the containing element instance is itself contained within an element containing a (possibly defaulted) SOAP-ENC:arrayType attribute or (c) or the name of the element bears a definite relation to the type, that type then determinable from a schema.
3. A simple value is represented as character data, that is, without any subelements. Every simple value must have a type that is either listed in the XML Schemas Specification, part 2 [11] or whose source type is listed therein (see also section 5.2).
4. A Compound Value is encoded as a sequence of elements, each accessor represented by an embedded element whose name corresponds to the name of the accessor. Accessors whose names are local to their containing types have unqualified element names; all others have qualified names (see also section 5.4).
5. A multi-reference simple or compound value is encoded as an independent element containing a local, unqualified attribute named "id" and of type "ID" per the XML Specification [7]. Each accessor to this value is an empty element having a local, unqualified attribute named "href"

and of type "uri-reference" per the XML Schema Specification [11], with a "href" attribute value of a URI fragment identifier referencing the corresponding independent element.

6. Strings and byte arrays are represented as multi-reference simple types, but special rules allow them to be represented efficiently for common cases (see also section 5.2.1 and 5.2.3). An accessor to a string or byte-array value MAY have an attribute named "id" and of type "ID" per the XML Specification [7]. If so, all other accessors to the same value are encoded as empty elements having a local, unqualified attribute named "href" and of type "uri-reference" per the XML Schema Specification [11], with a "href" attribute value of a URI fragment identifier referencing the single element containing the value.
7. It is permissible to encode several references to a value as though these were references to several distinct values, but only when from context it is known that the meaning of the XML instance is unaltered.
8. Arrays are compound values (see also section 5.4.2). SOAP arrays are defined as having a type of "SOAP-ENC:Array" or a type derived there from.

   SOAP arrays have one or more dimensions (rank) whose members are distinguished by ordinal position. An array value is represented as a series of elements reflecting the array, with members appearing in ascending ordinal sequence. For multi-dimensional arrays the dimension on the right side varies most rapidly. Each member element is named as an independent element (see rule 2).

   SOAP arrays can be single-reference or multi-reference values, and consequently may be represented as the content of either an embedded or independent element.

   SOAP arrays MUST contain a "SOAP-ENC:arrayType" attribute whose value specifies the type of the contained elements as well as the dimension(s) of the array. The value of the "SOAP-ENC:arrayType" attribute is defined as follows:

```
arrayTypeValue = atype asize
atype          = QName *( rank )
rank           = "[" *( "," ) "]"
asize          = "[" #length "]"
length         = 1*DIGIT
```

   The "atype" construct is the type name of the contained elements expressed as a QName as would appear in the "type" attribute of an XML Schema element declaration and acts as a type constraint (meaning that all values of contained elements are asserted to conform to the indicated type; that is, the type cited in SOAP-ENC:arrayType must be the type or a supertype of every array member). In the case of arrays of arrays or "jagged arrays", the type component is encoded as the "innermost" type name followed by a rank construct for each level of nested arrays starting from 1. Multi-dimensional arrays are encoded using a comma for each dimension starting from 1.

   The "asize" construct contains a comma separated list of zero, one, or more integers indicating the lengths of each dimension of the array. A value of zero integers indicates that no particular quantity is asserted but that the size may be determined by inspection of the actual members.

   For example, an array with 5 members of type array of integers would have an arrayTypeValue

value of "int[][5]" of which the atype value is "int[]" and the asize value is "[5]". Likewise, an array with 3 members of type two-dimensional arrays of integers would have an arrayTypeValue value of "int[,][3]" of which the atype value is "int[,]" and the asize value is "[3]".

A SOAP array member MAY contain a "SOAP-ENC:offset" attribute indicating the offset position of that item in the enclosing array. This can be used to indicate the offset position of a partially represented array (see section 5.4.2.1). Likewise, an array member MAY contain a "SOAP-ENC:position" attribute indicating the position of that item in the enclosing array. This can be used to describe members of sparse arrays (see section 5.4.2.2). The value of the "SOAP-ENC:offset" and the "SOAP-ENC:position" attribute is defined as follows:

```
    arrayPoint = "[" #length "]"
```

with offsets and positions based at 0.

9. A NULL value or a default value MAY be represented by omission of the accessor element. A NULL value MAY also be indicated by an accessor element containing the attribute xsi:null with value '1' or possibly other application-dependent attributes and values.

Note that rule 2 allows independent elements and also elements representing the members of arrays to have names which are not identical to the type of the contained value.

5.2 Simple Types

For simple types, SOAP adopts all the types found in the section "Built-in datatypes" of the "XML Schema Part 2: Datatypes" Specification [11], both the value and lexical spaces. Examples include:

Type	Example
int	58502
float	314159265358979E+1
negativeInteger	-32768
string	Louis "Satchmo" Armstrong

The datatypes declared in the XML Schema specification may be used directly in element schemas. Types derived from these may also be used. An example of a schema fragment and corresponding instance data with elements of these types is:

```
<element name="age" type="int"/>
<element name="height" type="float"/>
<element name="displacement" type="negativeInteger"/>
<element name="color">
  <simpleType base="xsd:string">
    <enumeration value="Green"/>
    <enumeration value="Blue"/>
  </simpleType>
```

```
</element>

<age>45</age>
<height>5.9</height>
<displacement>-450</displacement>
<color>Blue</color>
```

All simple values MUST be encoded as the content of elements whose type is either defined in "XML Schema Part 2: Datatypes" Specification [11], or is based on a type found there by using the mechanisms provided in the XML Schema specification.

If a simple value is encoded as an independent element or member of a heterogenous array it is convenient to have an element declaration corresponding to the datatype. Because the "XML Schema Part 2: Datatypes" Specification [11] includes type definitions but does not include corresponding element declarations, the SOAP-ENC schema and namespace declares an element for every simple datatype. These MAY be used.

```
<SOAP-ENC:int id="int1">45</SOAP-ENC:int>
```

5.2.1 Strings

The datatype "string" is defined in "XML Schema Part 2: Datatypes" Specification [11]. Note that this is not identical to the type called "string" in many database or programming languages, and in particular may forbid some characters those languages would permit. (Those values must be represented by using some datatype other than xsd:string.)

A string MAY be encoded as a single-reference or a multi-reference value.

The containing element of the string value MAY have an "id" attribute. Additional accessor elements MAY then have matching "href" attributes.

For example, two accessors to the same string could appear, as follows:

```
<greeting id="String-0">Hello</greeting>
<salutation href="#String-0"/>
```

However, if the fact that both accessors reference the same instance of the string (or subtype of string) is immaterial, they may be encoded as two single-reference values as follows:

```
<greeting>Hello</greeting>
<salutation>Hello</salutation>
```

Schema fragments for these examples could appear similar to the following:

```
<element name="greeting" type="SOAP-ENC:string"/>
<element name="salutation" type="SOAP-ENC:string"/>
```

(In this example, the type SOAP-ENC:string is used as the element's type as a convenient way to declare an element whose datatype is "xsd:string" and which also allows an "id" and "href" attribute. See the SOAP Encoding schema for the exact definition. Schemas MAY use these declarations from the SOAP Encoding schema but are not required to.)

5.2.2 Enumerations

The "XML Schema Part 2: Datatypes" Specification [11] defines a mechanism called "enumeration." The SOAP data model adopts this mechanism directly. However, because programming and other languages often define enumeration somewhat differently, we spell-out the concept in more detail here and describe how a value that is a member of an enumerated list of possible values is to be encoded. Specifically, it is encoded as the name of the value.

"Enumeration" as a concept indicates a set of distinct names. A specific enumeration is a specific list of distinct values appropriate to the base type. For example the set of color names ("Green", "Blue", "Brown") could be defined as an enumeration based on the string built-in type. The values ("1", "3", "5") are a possible enumeration based on integer, and so on. "XML Schema Part 2: Datatypes" [11] supports enumerations for all of the simple types except for boolean. The language of "XML Schema Part 1: Structures" Specification [10] can be used to define enumeration types. If a schema is generated from another notation in which no specific base type is applicable, use "string". In the following schema example "EyeColor" is defined as a string with the possible values of "Green", "Blue", or "Brown" enumerated, and instance data is shown accordingly.

```
<element name="EyeColor" type="tns:EyeColor"/>
<simpleType name="EyeColor" base="xsd:string">
   <enumeration value="Green"/>
   <enumeration value="Blue"/>
   <enumeration value="Brown"/>
</simpleType>

<Person>
   <Name>Henry Ford</Name>
   <Age>32</Age>
   <EyeColor>Brown</EyeColor>
</Person>
```

5.2.3 Array of Bytes

An array of bytes MAY be encoded as a single-reference or a multi-reference value. The rules for an array of bytes are similar to those for a string.

In particular, the containing element of the array of bytes value MAY have an "id" attribute. Additional accessor elements MAY then have matching "href" attributes.

The recommended representation of an opaque array of bytes is the 'base64' encoding defined in XML Schemas [10][11], which uses the base64 encoding algorithm defined in 2045 [13]. However, the line length restrictions that normally apply to base64 data in MIME do not apply in SOAP. A "SOAP-ENC:base64" subtype is supplied for use with SOAP.

```
<picture xsi:type="SOAP-ENC:base64">
   aG93IG5vDyBicm73biBjb3cNCg==
</picture>
```

5.3 Polymorphic Accessor

Many languages allow accessors that can polymorphically access values of several types, each type being available at run time. A polymorphic accessor instance MUST contain an "xsi:type" attribute that describes the type of the actual value.

For example, a polymorphic accessor named "cost" with a value of type "xsd:float" would be encoded as follows:

```
<cost xsi:type="xsd:float">29.95</cost>
```

as contrasted with a cost accessor whose value's type is invariant, as follows:

```
<cost>29.95</cost>
```

5.4 Compound types

SOAP defines types corresponding to the following structural patterns often found in programming languages:

Struct
: A "struct" is a compound value in which accessor name is the only distinction among member values, and no accessor has the same name as any other.

Array
: An "array" is a compound value in which ordinal position serves as the only distinction among member values.

SOAP also permits serialization of data that is neither a Struct nor an Array, for example data such as is found in a Directed-Labeled-Graph Data Model in which a single node has many distinct accessors, some of which occur more than once. SOAP serialization does not require that the underlying data model make an ordering distinction among accessors, but if such an order exists, the accessors MUST be encoded in that sequence.

5.4.1 Compound Values, Structs and References to Values

The members of a Compound Value are encoded as accessor elements. When accessors are distinguished by their name (as for example in a struct), the accessor name is used as the element name. Accessors whose names are local to their containing types have unqualified element names; all others have qualified names.

The following is an example of a struct of type "Book":

```
<e:Book>
   <author>Henry Ford</author>
   <preface>Prefatory text</preface>
   <intro>This is a book.</intro>
</e:Book>
```

And this is a schema fragment describing the above structure:

```
<element name="Book">
<complexType>
  <element name="author" type="xsd:string"/>
  <element name="preface" type="xsd:string"/>
   <element name="intro" type="xsd:string"/>
</complexType>
</e:Book>
```

Below is an example of a type with both simple and complex members. It shows two levels of referencing. Note that the "href" attribute of the "Author" accessor element is a reference to the value whose "id" attribute matches. A similar construction appears for the "Address".

```
<e:Book>
   <title>My Life and Work</title>
   <author href="#Person-1"/>
</e:Book>
<e:Person id="Person-1">
   <name>Henry Ford</name>
   <address href="#Address-2"/>
</e:Person>
<e:Address id="Address-2">
   <email>mailto:henryford@hotmail.com</email>
   <web>http://www.henryford.com</web>
</e:Address>
```

The form above is appropriate when the "Person" value and the "Address" value are multi-reference. If these were instead both single-reference, they SHOULD be embedded, as follows:

```
<e:Book>
   <title>My Life and Work</title>
   <author>
       <name>Henry Ford</name>
       <address>
          <email>mailto:henryford@hotmail.com</email>
          <web>http://www.henryford.com</web>
       </address>
   </author>
</e:Book>
```

If instead there existed a restriction that no two persons can have the same address in a given instance and that an address can be either a Street-address or an Electronic-address, a Book with two authors would be encoded as follows:

```
<e:Book>
   <title>My Life and Work</title>
   <firstauthor href="#Person-1"/>
   <secondauthor href="#Person-2"/>
</e:Book>
```

```
<e:Person id="Person-1">
   <name>Henry Ford</name>
   <address xsi:type="m:Electronic-address">
       <email>mailto:henryford@hotmail.com</email>
       <web>http://www.henryford.com</web>
   </address>
</e:Person>
<e:Person id="Person-2">
   <name>Samuel Crowther</name>
   <address xsi:type="n:Street-address">
       <street>Martin Luther King Rd</street>
       <city>Raleigh</city>
       <state>North Carolina</state>
   </address>
</e:Person>
```

Serializations can contain references to values not in the same resource:

```
<e:Book>
   <title>Paradise Lost</title>
   <firstauthor href="http://www.dartmouth.edu/~milton/"/>
</e:Book>
```

And this is a schema fragment describing the above structures:

```
<element name="Book" type="tns:Book"/>
<complexType name="Book">
   <!-- Either the following group must occur or else the
        href attribute must appear, but not both. -->
   <sequence minOccurs="0" maxOccurs="1">
       <element name="title" type="xsd:string"/>
       <element name="firstauthor" type="tns:Person"/>
       <element name="secondauthor" type="tns:Person"/>
   </sequence>
   <attribute name="href" type="uriReference"/>
   <attribute name="id" type="ID"/>
   <anyAttribute namespace="##other"/>
</complexType>

<element name="Person" base="tns:Person"/>
<complexType name="Person">
   <!-- Either the following group must occur or else the
        href attribute must appear, but not both. -->
   <sequence minOccurs="0" maxOccurs="1">
       <element name="name" type="xsd:string"/>
       <element name="address" type="tns:Address"/>
   </sequence>
   <attribute name="href" type="uriReference"/>
   <attribute name="id" type="ID"/>
   <anyAttribute namespace="##other"/>
```

```
</complexType>

<element name="Address" base="tns:Address"/>
<complexType name="Address">
   <!-- Either the following group must occur or else the
        href attribute must appear, but not both. -->
   <sequence minOccurs="0" maxOccurs="1">
       <element name="street" type="xsd:string"/>
       <element name="city" type="xsd:string"/>
       <element name="state" type="xsd:string"/>
   </sequence>
   <attribute name="href" type="uriReference"/>
   <attribute name="id" type="ID"/>
   <anyAttribute namespace="##other"/>
</complexType>
```

5.4.2 Arrays

SOAP arrays are defined as having a type of "SOAP-ENC:Array" or a type derived there from (see also rule 8). Arrays are represented as element values, with no specific constraint on the name of the containing element (just as values generally do not constrain the name of their containing element).

Arrays can contain elements which themselves can be of any type, including nested arrays. New types formed by restrictions of SOAP-ENC:Array can also be created to represent, for example, arrays limited to integers or arrays of some user-defined enumeration.

The representation of the value of an array is an ordered sequence of elements constituting the items of the array. Within an array value, element names are not significant for distinguishing accessors. Elements may have any name. In practice, elements will frequently be named so that their declaration in a schema suggests or determines their type. As with compound types generally, if the value of an item in the array is a single-reference value, the item contains its value. Otherwise, the item references its value via an "href" attribute.

The following example is a schema fragment and an array containing integer array members.

```
<element name="myFavoriteNumbers"
        type="SOAP-ENC:Array"/>

<myFavoriteNumbers
  SOAP-ENC:arrayType="xsd:int[2]">
   <number>3</number>
   <number>4</number>
</myFavoriteNumbers>
```

In that example, the array "myFavoriteNumbers" contains several members each of which is a value of type SOAP-ENC:int. This can be determined by inspection of the SOAP-ENC:arrayType attribute. Note that the SOAP-ENC:Array type allows unqualified element names without restriction. These convey no type information, so when used they must either have an xsi:type attribute or the containing element must have a SOAP-ENC:arrayType attribute. Naturally, types derived from SOAP-ENC:Array may declare local elements, with type information.

As previously noted, the SOAP-ENC schema contains declarations of elements with names corresponding to each simple type in the "XML Schema Part 2: Datatypes" Specification [11]. It also contains a declaration for "Array". Using these, we might write

```
<SOAP-ENC:Array SOAP-ENC:arrayType="xsd:int[2]">
   <SOAP-ENC:int>3</SOAP-ENC:int>
   <SOAP-ENC:int>4</SOAP-ENC:int>
</SOAP-ENC:Array>
```

Arrays can contain instances of any subtype of the specified arrayType. That is, the members may be of any type that is substitutable for the type specified in the arrayType attribute, according to whatever substitutability rules are expressed in the schema. So, for example, an array of integers can contain any type derived from integer (for example "int" or any user-defined derivation of integer). Similarly, an array of "address" might contain a restricted or extended type such as "internationalAddress". Because the supplied SOAP-ENC:Array type admits members of any type, arbitrary mixtures of types can be contained unless specifically limited by use of the arrayType attribute.

Types of member elements can be specified using the xsi:type attribute in the instance, or by declarations in the schema of the member elements, as the following two arrays demonstrate respectively.

```
<SOAP-ENC:Array SOAP-ENC:arrayType="xsd:ur-type[4]">
   <thing xsi:type="xsd:int">12345</thing>
   <thing xsi:type="xsd:decimal">6.789</thing>
   <thing xsi:type="xsd:string">
      Of Mans First Disobedience, and the Fruit
      Of that Forbidden Tree, whose mortal tast
      Brought Death into the World, and all our woe,
   </thing>
   <thing xsi:type="xsd:uriReference">
      http://www.dartmouth.edu/~milton/reading_room/
   </thing>
</SOAP-ENC:Array>

<SOAP-ENC:Array SOAP-ENC:arrayType="xsd:ur-type[4]">
   <SOAP-ENC:int>12345</SOAP-ENC:int>
   <SOAP-ENC:decimal>6.789</SOAP-ENC:decimal>
   <xsd:string>
      Of Mans First Disobedience, and the Fruit
      Of that Forbidden Tree, whose mortal tast
      Brought Death into the World, and all our woe,
   </xsd:string>
   <SOAP-ENC:uriReference>
      http://www.dartmouth.edu/~milton/reading_room/
   </SOAP-ENC:uriReference >
</SOAP-ENC:Array>
```

Array values may be structs or other compound values. For example an array of "xyz:Order" structs :

```
<SOAP-ENC:Array SOAP-ENC:arrayType="xyz:Order[2]">
   <Order>
       <Product>Apple</Product>
       <Price>1.56</Price>
   </Order>
   <Order>
       <Product>Peach</Product>
       <Price>1.48</Price>
   </Order>
</SOAP-ENC:Array>
```

Arrays may have other arrays as member values. The following is an example of an array of two arrays, each of which is an array of strings.

```
<SOAP-ENC:Array SOAP-ENC:arrayType="xsd:string[][2]">
   <item href="#array-1"/>
   <item href="#array-2"/>
</SOAP-ENC:Array>
<SOAP-ENC:Array id="array-1" SOAP-ENC:arrayType="xsd:string[2]">
   <item>r1c1</item>
   <item>r1c2</item>
   <item>r1c3</item>
</SOAP-ENC:Array>
<SOAP-ENC:Array id="array-2" SOAP-ENC:arrayType="xsd:string[2]">
   <item>r2c1</item>
   <item>r2c2</item>
</SOAP-ENC:Array>
```

The element containing an array value does not need to be named "SOAP-ENC:Array". It may have any name, provided that the type of the element is either SOAP-ENC:Array or is derived from SOAP-ENC:Array by restriction. For example, the following is a fragment of a schema and a conforming instance array.

```
<simpleType name="phoneNumber" base="string"/>

<element name="ArrayOfPhoneNumbers">
  <complexType base="SOAP-ENC:Array">
    <element name="phoneNumber" type="tns:phoneNumber"
maxOccurs="unbounded"/>
  </complexType>
  <anyAttribute/>
</element>

<xyz:ArrayOfPhoneNumbers SOAP-ENC:arrayType="xyz:phoneNumber[2]">
   <phoneNumber>206-555-1212</phoneNumber>
   <phoneNumber>1-888-123-4567</phoneNumber>
</xyz:ArrayOfPhoneNumbers>
```

Arrays may be multi-dimensional. In this case, more than one size will appear within the asize part of the arrayType attribute:

```
<SOAP-ENC:Array SOAP-ENC:arrayType="xsd:string[2,3]">
   <item>r1c1</item>
   <item>r1c2</item>
   <item>r1c3</item>
   <item>r2c1</item>
   <item>r2c2</item>
   <item>r2c3</item>
</SOAP-ENC:Array>
```

While the examples above have shown arrays encoded as independent elements, array values MAY also appear embedded and SHOULD do so when they are known to be single reference.

The following is an example of a schema fragment and an array of phone numbers embedded in a struct of type "Person" and accessed through the accessor "phone-numbers":

```
<simpleType name="phoneNumber" base="string"/>

<element name="ArrayOfPhoneNumbers">
  <complexType base="SOAP-ENC:Array">
    <element name="phoneNumber" type="tns:phoneNumber"
maxOccurs="unbounded"/>
  </complexType>
  <anyAttribute/>
</element>

<element name="Person">
  <complexType>
    <element name="name" type="string"/>
    <element name="phoneNumbers" type="tns:ArrayOfPhoneNumbers"/>
  </complexType>
</element>

<xyz:Person>
   <name>John Hancock</name>
   <phoneNumbers SOAP-ENC:arrayType="xyz:phoneNumber[2]">
       <phoneNumber>206-555-1212</phoneNumber>
       <phoneNumber>1-888-123-4567</phoneNumber>
   </phoneNumbers>
</xyz:Person>
```

Here is another example of a single-reference array value encoded as an embedded element whose containing element name is the accessor name:

```
<xyz:PurchaseOrder>
   <CustomerName>Henry Ford</CustomerName>
   <ShipTo>
       <Street>5th Ave</Street>
       <City>New York</City>
       <State>NY</State>
       <Zip>10010</Zip>
```

```
      </ShipTo>
      <PurchaseLineItems SOAP-ENC:arrayType="Order[2]">
          <Order>
              <Product>Apple</Product>
              <Price>1.56</Price>
          </Order>
          <Order>
              <Product>Peach</Product>
              <Price>1.48</Price>
          </Order>
      </PurchaseLineItems>
   </xyz:PurchaseOrder>
```

5.4.2.1 Partially Transmitted Arrays

SOAP provides support for partially transmitted arrays, known as "varying" arrays in some contexts [12]. A partially transmitted array indicates in an "SOAP-ENC:offset" attribute the zero-origin offset of the first element transmitted. If omitted, the offset is taken as zero.

The following is an example of an array of size five that transmits only the third and fourth element counting from zero:

```
<SOAP-ENC:Array SOAP-ENC:arrayType="xsd:string[5]" SOAP-
ENC:offset="[2]">
   <item>The third element</item>
   <item>The fourth element</item>
</SOAP-ENC:Array>
```

5.4.2.2 Sparse Arrays

SOAP provides support for sparse arrays. Each element representing a member value contains a "SOAP-ENC:position" attribute that indicates its position within the array. The following is an example of a sparse array of two-dimensional arrays of strings. The size is 4 but only position 2 is used:

```
<SOAP-ENC:Array SOAP-ENC:arrayType="xsd:string[,][4]">
   <SOAP-ENC:Array href="#array-1" SOAP-ENC:position="[2]"/>
</SOAP-ENC:Array>
<SOAP-ENC:Array id="array-1" SOAP-ENC:arrayType="xsd:string
[10,10]">
   <item SOAP-ENC:position="[2,2]">Third row, third col</item>
   <item SOAP-ENC:position="[7,2]">Eighth row, third col</item>
</SOAP-ENC:Array>
```

If the only reference to array-1 occurs in the enclosing array, this example could also have been encoded as follows:

```
<SOAP-ENC:Array SOAP-ENC:arrayType="xsd:string[,][4]">
  <SOAP-ENC:Array SOAP-ENC:position="[2]" SOAP-
ENC:arrayType="xsd:string[10,10]>
```

```
    <item SOAP-ENC:position="[2,2]">Third row, third col</item>
    <item SOAP-ENC:position="[7,2]">Eighth row, third col</item>
  </SOAP-ENC:Array>
</SOAP-ENC:Array>
```

5.4.3 Generic Compound Types

The encoding rules just cited are not limited to those cases where the accessor names are known in advance. If accessor names are known only by inspection of the immediate values to be encoded, the same rules apply, namely that the accessor is encoded as an element whose name matches the name of the accessor, and the accessor either contains or references its value. Accessors containing values whose types cannot be determined in advance MUST always contain an appropriate xsi:type attribute giving the type of the value.

Similarly, the rules cited are sufficient to allow serialization of compound types having a mixture of accessors distinguished by name and accessors distinguished by both name and ordinal position. (That is, having some accessors repeated.) This does not require that any schema actually contain such types, but rather says that if a type-model schema does have such types, a corresponding XML syntactic schema and instance may be generated.

```
<xyz:PurchaseOrder>
   <CustomerName>Henry Ford</CustomerName>
   <ShipTo>
       <Street>5th Ave</Street>
       <City>New York</City>
       <State>NY</State>
       <Zip>10010</Zip>
   </ShipTo>
   <PurchaseLineItems>
       <Order>
           <Product>Apple</Product>
           <Price>1.56</Price>
       </Order>
       <Order>
           <Product>Peach</Product>
           <Price>1.48</Price>
       </Order>
   </PurchaseLineItems>
</xyz:PurchaseOrder>
```

Similarly, it is valid to serialize a compound value that structurally resembles an arrray but is not of type (or subtype) SOAP-ENC:Array. For example:

```
<PurchaseLineItems>
    <Order>
        <Product>Apple</Product>
        <Price>1.56</Price>
    </Order>
    <Order>
        <Product>Peach</Product>
```

```
            <Price>1.48</Price>
        </Order>
    </PurchaseLineItems>
```

5.5 Default Values

An omitted accessor element implies either a default value or that no value is known. The specifics depend on the accessor, method, and its context. For example, an omitted accessor typically implies a Null value for polymorphic accessors (with the exact meaning of Null accessor-dependent). Likewise, an omitted Boolean accessor typically implies either a False value or that no value is known, and an omitted numeric accessor typically implies either that the value is zero or that no value is known.

5.6 SOAP root Attribute

The SOAP root attribute can be used to label serialization roots that are not true roots of an object graph so that the object graph can be deserialized. The attribute can have one of two values, either "1" or "0". True roots of an object graph have the implied attribute value of "1". Serialization roots that are not true roots can be labeled as serialization roots with an attribute value of "1" An element can explicitly be labeled as not being a serialization root with a value of "0".

The SOAP root attribute MAY appear on any subelement within the SOAP Header and SOAP Body elements. The attribute does not have a default value.

6. Using SOAP in HTTP

This section describes how to use SOAP within HTTP with or without using the HTTP Extension Framework. Binding SOAP to HTTP provides the advantage of being able to use the formalism and decentralized flexibility of SOAP with the rich feature set of HTTP. Carrying SOAP in HTTP does not mean that SOAP overrides existing semantics of HTTP but rather that the semantics of SOAP over HTTP maps naturally to HTTP semantics.

SOAP naturally follows the HTTP request/response message model providing SOAP request parameters in a HTTP request and SOAP response parameters in a HTTP response. Note, however, that SOAP intermediaries are NOT the same as HTTP intermediaries. That is, an HTTP intermediary addressed with the HTTP Connection header field cannot be expected to inspect or process the SOAP entity body carried in the HTTP request.

HTTP applications MUST use the media type "text/xml" according to RFC 2376 [3] when including SOAP entity bodies in HTTP messages.

6.1 SOAP HTTP Request

Although SOAP might be used in combination with a variety of HTTP request methods, this binding only defines SOAP within HTTP POST requests (see section 7 for how to use SOAP for RPC and section 6.3 for how to use the HTTP Extension Framework).

6.1.1 The SOAPAction HTTP Header Field

The SOAPAction HTTP request header field can be used to indicate the intent of the SOAP HTTP request. The value is a URI identifying the intent. SOAP places no restrictions on the format or specificity of the URI or that it is resolvable. An HTTP client MUST use this header field when issuing a SOAP HTTP Request.

```
soapaction    = "SOAPAction" ":" [ <"> URI-reference <"> ]
URI-reference = <as defined in RFC 2396 [4]>
```

The presence and content of the SOAPAction header field can be used by servers such as firewalls to appropriately filter SOAP request messages in HTTP. The header field value of empty string ("") means that the intent of the SOAP message is provided by the HTTP Request-URI. No value means that there is no indication of the intent of the message.

Examples:

```
SOAPAction: "http://electrocommerce.org/abc#MyMessage"
SOAPAction: "myapp.sdl"
SOAPAction: ""
SOAPAction:
```

6.2 SOAP HTTP Response

SOAP HTTP follows the semantics of the HTTP Status codes for communicating status information in HTTP. For example, a 2xx status code indicates that the client's request including the SOAP component was successfully received, understood, and accepted etc.

In case of a SOAP error while processing the request, the SOAP HTTP server MUST issue an HTTP 500 "Internal Server Error" response and include a SOAP message in the response containing a SOAP Fault element (see section 4.4) indicating the SOAP processing error.

6.3 The HTTP Extension Framework

A SOAP message MAY be used together with the HTTP Extension Framework [6] in order to identify the presence and intent of a SOAP HTTP request.

Whether to use the Extension Framework or plain HTTP is a question of policy and capability of the communicating parties. Clients can force the use of the HTTP Extension Framework by using a mandatory extension declaration and the "M-" HTTP method name prefix. Servers can force the use of the HTTP Extension Framework by using the 510 "Not Extended" HTTP status code. That is, using one extra round trip, either party can detect the policy of the other party and act accordingly.

The extension identifier used to identify SOAP using the Extension Framework is

```
http://schemas.xmlsoap.org/soap/envelope/
```

6.4 SOAP HTTP Examples

Example 3 SOAP HTTP Using POST

```
POST /StockQuote HTTP/1.1
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "http://electrocommerce.org/abc#MyMessage"

<SOAP-ENV:Envelope...

HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope...
```

Example 4 SOAP Using HTTP Extension Framework

```
M-POST /StockQuote HTTP/1.1
Man: "http://schemas.xmlsoap.org/soap/envelope/"; ns=NNNN
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
NNNN-SOAPAction: "http://electrocommerce.org/abc#MyMessage"

<SOAP-ENV:Envelope...

HTTP/1.1 200 OK
Ext:
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope...
```

7. Using SOAP for RPC

One of the design goals of SOAP is to encapsulate and exchange RPC calls using the extensibility and flexibility of XML. This section defines a uniform representation of remote procedure calls and responses.

Although it is anticipated that this representation is likely to be used in combination with the encoding style defined in section 5 other representations are possible. The SOAP encodingStyle attribute (see section 4.3.2) can be used to indicate the encoding style of the method call and or the response using the representation described in this section.

Using SOAP for RPC is orthogonal to the SOAP protocol binding (see section 6). In the case of using HTTP as the protocol binding, an RPC call maps naturally to an HTTP request and an RPC response maps to an HTTP response. However, using SOAP for RPC is not limited to the HTTP protocol binding.

To make a method call, the following information is needed:

- The URI of the target object
- A method name
- An optional method signature
- The parameters to the method
- Optional header data

SOAP relies on the protocol binding to provide a mechanism for carrying the URI. For example, for HTTP the request URI indicates the resource that the invocation is being made against. Other than it be a valid URI, SOAP places no restriction on the form of an address (see [4] for more information on URIs).

7.1 RPC and SOAP Body

RPC method calls and responses are both carried in the SOAP Body element (see section 4.3) using the following representation:

- A method invocation is modelled as a struct.
- The method invocation is viewed as a single struct containing an accessor for each [in] or [in/out] parameter. The struct is both named and typed identically to the method name.
- Each [in] or [in/out] parameter is viewed as an accessor, with a name corresponding to the name of the parameter and type corresponding to the type of the parameter. These appear in the same order as in the method signature.
- A method response is modelled as a struct.
- The method response is viewed as a single struct containing an accessor for the return value and each [out] or [in/out] parameter. The first accessor is the return value followed by the parameters in the same order as in the method signature.
- Each parameter accessor has a name corresponding to the name of the parameter and type corresponding to the type of the parameter. The name of the return value accessor is not significant. Likewise, the name of the struct is not significant. However, a convention is to name it after the method name with the string "Response" appended.
- A method fault is encoded using the SOAP Fault element (see section 4.4). If a protocol binding adds additional rules for fault expression, those also MUST be followed.

As noted above, method and response structs can be encoded according to the rules in section 5, or other encodings can be specified using the encodingStyle attribute (see section 4.1.1).

Applications MAY process requests with missing parameters but also MAY return a fault.

Because a result indicates success and a fault indicates failure, it is an error for the method response to contain both a result and a fault.

7.2 RPC and SOAP Header

Additional information relevant to the encoding of a method request but not part of the formal method signature MAY be expressed in the RPC encoding. If so, it MUST be expressed as a subelement of the SOAP Header element.

An example of the use of the header element is the passing of a transaction ID along with a message. Since the transaction ID is not part of the signature and is typically held in an infrastructure component rather than application code, there is no direct way to pass the necessary information with the call. By adding an entry to the headers and giving it a fixed name, the transaction manager on the receiving side can extract the transaction ID and use it without affecting the coding of remote procedure calls.

8. Security Considerations

Not described in this document are methods for integrity and privacy protection. Such issues will be addressed more fully in a future version(s) of this document.

9. References

[1] S. Bradner, "The Internet Standards Process -- Revision 3", RFC2026, Harvard University, October 1996

[2] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, Harvard University, March 1997

[3] E. Whitehead, M. Murata, "XML Media Types", RFC2376, UC Irvine, Fuji Xerox Info. Systems, July 1998

[4] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998.

[5] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, U.C. Irvine, DEC W3C/MIT, DEC, W3C/MIT, W3C/MIT, January 1997

[6] H. Nielsen, P. Leach, S. Lawrence, "An HTTP Extension Framework", RFC 2774, Microsoft, Microsoft, Agranat Systems

[7] W3C Recommendation "The XML Specification"

[8] W3C Recommendation "Namespaces in XML"

[9] W3C Working Draft "XML Linking Language". This is work in progress.

[10] W3C Working Draft "XML Schema Part 1: Structures". This is work in progress.

[11] W3C Working Draft "XML Schema Part 2: Datatypes". This is work in progress.

[12] Transfer Syntax NDR, in "DCE 1.1: Remote Procedure Call"

[13] N. Freed, N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC2045, Innosoft, First Virtual, November 1996

A. SOAP Envelope Examples

A.1 Sample Encoding of Call Requests

Example 5 Similar to Example 1 but with a Mandatory Header

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   <SOAP-ENV:Header>
       <t:Transaction
           xmlns:t="some-URI"
           SOAP-ENV:mustUnderstand="1">
               5
       </t:Transaction>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
       <m:GetLastTradePrice xmlns:m="Some-URI">
           <symbol>DEF</symbol>
       </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example 6 Similar to Example 1 but with multiple request parameters

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
```

```
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   <SOAP-ENV:Body>
       <m:GetLastTradePriceDetailed
         xmlns:m="Some-URI">
           <Symbol>DEF</Symbol>
           <Company>DEF Corp</Company>
           <Price>34.1</Price>
       </m:GetLastTradePriceDetailed>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

A.2 Sample Encoding of Response

Example 7 Similar to Example 2 but with a Mandatory Header

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   <SOAP-ENV:Header>
       <t:Transaction
         xmlns:t="some-URI"
         xsi:type="xsd:int" mustUnderstand="1">
           5
       </t:Transaction>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
       <m:GetLastTradePriceResponse
         xmlns:m="Some-URI">
           <Price>34.5</Price>
       </m:GetLastTradePriceResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example 8 Similar to Example 2 but with a Struct

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-
```

```
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   <SOAP-ENV:Body>
       <m:GetLastTradePriceResponse
         xmlns:m="Some-URI">
           <PriceAndVolume>
               <LastTradePrice>
                   34.5
               </LastTradePrice>
               <DayVolume>
                   10000
               </DayVolume>
           </PriceAndVolume>
       </m:GetLastTradePriceResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example 9 Similar to Example 2 but Failing to honor Mandatory Header

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Body>
       <SOAP-ENV:Fault>
           <faultcode>SOAP-ENV:MustUnderstand</faultcode>
           <faultstring>SOAP Must Understand Error</faultstring>
       </SOAP-ENV:Fault>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example 10 Similar to Example 2 but Failing to handle Body

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Body>
       <SOAP-ENV:Fault>
           <faultcode>SOAP-ENV:Server</faultcode>
           <faultstring>Server Error</faultstring>
           <detail>
               <e:myfaultdetails xmlns:e="Some-URI">
                 <message>
                   My application didn't work
                 </message>
                 <errorcode>
```

```
                    1001
                  </errorcode>
                </e:myfaultdetails>
            </detail>
        </SOAP-ENV:Fault>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

APPENDIX D

A Gentle Introduction to Mutual Recursion

Manuel Rubio-Sánchez
Departamento de Lenguajes y Sistemas Informáticos 1
Universidad Rey Juan Carlos
c/ Tulipán s/n, 28933 Móstoles
Madrid, Spain
manuel.rubio@urjc.es

Jaime Urquiza-Fuentes
Departamento de Lenguajes y Sistemas Informáticos 1
Universidad Rey Juan Carlos
c/ Tulipán s/n, 28933 Móstoles
Madrid, Spain
jaime.urquiza@urjc.es

Cristóbal Pareja-Flores
Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid
Avda. Puerta de Hierro s/n
28040 Madrid, Spain
cpareja@sip.ucm.es

ABSTRACT

Recursion is an important topic in computer science curricula. It is related to the acquisition of competences regarding problem decomposition, functional abstraction and the concept of induction. In comparison with direct recursion, mutual recursion is considered to be more complex. Consequently, it is generally addressed superficially in CS1/2 programming courses and textbooks. We show that, when a problem is approached appropriately, not only can mutual recursion be a powerful tool, but it can also be easy to understand and fun. This paper provides several intuitive and attractive algorithms that rely on mutual recursion, and which have been designed to help strengthen students' ability to decompose problems and apply induction. Furthermore, we show that a solution based on mutual recursion may be easier to design, prove and comprehend than other solutions based on direct recursion. We have evaluated the use of these algorithms while teaching recursion concepts. Results suggest that mutual recursion, in comparison with other types of recursion, is not as hard as it seems when: (1) determining the result of a (mathematical) function call, and, most importantly, (2) designing algorithms for solving simple problems.

Categories and Subject Descriptors

D.1.1 [**Applicative (Functional) Programming**]: Recursion; G.2.1 [**Combinatorics**]: Counting problems; K.3.2 [**Computer and Information Science Education**]: Computer science education

General Terms

Algorithms

Keywords

Mutual recursion, recursion problems, combinatorics, counting problems, Fibonacci numbers.

1. TEACHING MUTUAL RECURSION

Recursion is a fundamental programming concept, and is therefore necessary in a computer science curriculum. It plays an important role in the acquisition of competences regarding functional abstraction and problem decomposition through the concept of induction. Nevertheless, computer science students generally find recursion hard to master. Numerous authors have tried to identify the factors (conceptual models, cognitive learning styles, functional abstraction reasoning) responsible for these difficulties [14, 12]. Others have proposed methods designed to overcome these problems by using visualizations of the recursion or activation tree [4, 9], or through three-dimensional animations [2]. Additional studies propose teaching methodologies, such as presenting concepts in gradual steps (firstly grammars, then functional languages, and finally recursion in imperative languages) [13].

Recursive algorithms can be categorized according to the number and type of the recursive function calls. A common classification distinguishes the following types: linear (of which tail recursion is a special case), multiple (or exponential), nested, and mutual. The last two types are considered to be especially complex, and, consequently, are generally not treated in depth in CS1/2 programming courses or textbooks. Moreover, they are usually explained by using a single basic example. Nested recursion is taught with the Ackermann function, whereas mutual recursion is seen through the following pair of functions for determining the parity of a nonnegative integer:

$$is\_odd(n) = \begin{cases} \text{FALSE} & \text{if } n = 0 \\ is\_even(n-1) & \text{if } n > 0 \end{cases}$$

$$is\_even(n) = \begin{cases} \text{TRUE} & \text{if } n = 0 \\ is\_odd(n-1) & \text{if } n > 0 \end{cases}$$

While nested recursion is rather infrequent, mutual recursion appears often in advanced computer science topics (e.g., parsers). This paper provides several simple, intuitive and attractive algorithms that rely on mutual recursion, the majority of which solve combinatorial problems involving Fibonacci numbers ($F_n = F_{n-1} + F_{n-2}$, $F_1 = 1$, $F_2 = 1$). The goal is to present algorithms that can help students focus on problem decomposition, induction and functional abstraction. Additionally, several of these algorithms provide solutions that may be easier to design, prove, and comprehend than other strategies based on direct recursion. The problems have been chosen carefully so that they can be implemented easily with imperative programming languages, avoiding issues related to mechanisms such as parameter passing, control stack, etc. (see [13]). A pedagogical evaluation suggests that mutual recursion, in comparison with direct recursion, is not as hard as it seems when: (1) determining the result of a (mathematical) function call, and, most importantly, (2) designing algorithms for solving simple problems.

The rest of the paper is organized as follows. Section 2 presents and analyzes the proposed combinatorial problems and their solutions. Section 3 shows the experimental results, while a discussion and conclusions are reported in Sec. 4.

2. COMBINATORIAL PROBLEMS

Recursion is often explained in CS1/2 courses with the aid of mathematical functions, such as the factorial, power, or binomial coefficient. An interesting characteristic shared by these functions is their relation with combinatorics (permutations, variations with repetition, combinations). However, most texts only focus on their recursive definitions, instead of providing (combinatorial) problems that can be decomposed recursively, and whose solutions exhibit the same structure inherent in the functions. For instance, it is easy to see that the number of different permutations f_n, of n different elements, can be decomposed as $f_n = n \cdot f_{n-1}$, with $f_0 = f_1 = 1$ ($f_n = n!$). Due to the abundance of basic combinatoric examples, there is a wide range of valuable combinatoric problems for teaching recursion [11].

Another rich class of combinatorial counting problems involves Fibonacci numbers. A collection of such problems can be found in [7]. Simpler ones can be decomposed easily into $f_n = f_{n-1}+f_{n-2}$, like the number of ways to climb a flight of n steps leaping one or two steps at a time ("Leonardo's leaps" problem). More challenging problems need other Fibonacci identities (e.g., involving the computation of the even or odd terms of the Fibonacci sequence, where $f_n = 3f_{n-1} - f_{n-2}$).

This section presents and analyzes solutions based on mutual recursion for several of these problems. The goal is to focus on problem decomposition and the concept of induction, rather than strictly concentrating on mathematical functions.

2.1 Fibonacci Rabbits Problem

The origin of the Fibonacci numbers and their sequence can be found in a thought experiment posed in 1202. It was related to the population growth of pairs of rabbits under ideal conditions. The objective was to calculate the size of a population of rabbits during each month according to the following rules:

- Originally, a newly-born pair of rabbits, one male and one female, are put in a field.
- Rabbits take one month to mature and never die.
- It takes one month for mature rabbits to produce another pair of newly-born rabbits, who will thereafter mate together.
- The female always produces one male and one female rabbit every month from the second month on.

2.1.1 Distinguishing Baby and Adult Pairs

This problem can be approached by considering baby and adult pairs separately [10]. Let B_i, A_i and T_i be the number of baby, adult and total number of pairs of rabbits, respectively, at month i. Since every adult pair produces a new pair of babies each month, the number of baby pairs at month i is equal to the number of adult pairs in the previous month $i - 1$:

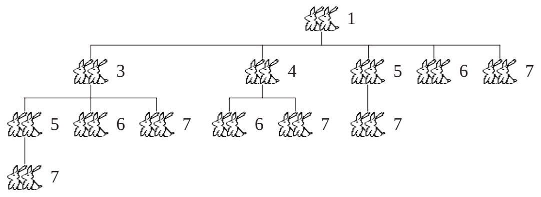


Figure 1: Genealogical tree of the rabbit population growth process.

$$B_i = \begin{cases} 1 & \text{if } i = 1 \\ A_{i-1} & \text{if } i \neq 1 \end{cases}$$

Furthermore, the number of adult pairs at month i is the sum of the adult pairs in the previous month $i - 1$, plus the number of baby pairs that will have matured over the previous month:

$$A_i = \begin{cases} 0 & \text{if } i = 1 \\ A_{i-1} + B_{i-1} & \text{if } i \neq 1 \end{cases}$$

The following identities can be easily proven:

$$T_i = A_i + B_i = A_{i+1} = B_{i+2} = F_i$$

Thus, A_i, B_i, and T_i are Fibonacci numbers.

The presented algorithm is a particular case of the following general mutually recursive functions:

$$\mathcal{B}_i = \begin{cases} \beta_{\rm b} & \text{if } i = 1 \\ \beta_{\rm r} + \mathcal{A}_{i-1} & \text{if } i \geq 2 \end{cases}$$

$$\mathcal{A}_i = \begin{cases} \alpha_{\rm b} & \text{if } i = 1 \\ \alpha_{\rm r} + \mathcal{A}_{i-1} + \mathcal{B}_{i-1} & \text{if } i \geq 2 \end{cases}$$

for $\beta_{\rm b} = 1$, y $\beta_{\rm r} = \alpha_{\rm b} = \alpha_{\rm r} = 0$. These functions are also related to Fibonacci numbers:

$$\mathcal{A}_n + \mathcal{B}_n + \alpha_{\rm r} + \beta_{\rm r} = \mathcal{A}_{n+1} + \beta_{\rm r} = \mathcal{B}_{n+2} =$$
$$= \beta_{\rm b} F_n + (\beta_{\rm r} + \alpha_{\rm b}) F_{n+1} + \alpha_{\rm r} F_{n+2} - \alpha_{\rm r}$$

Since each constant is associated to a specific type of node on the recursion tree, it is easy to see that the cardinality of many subsets of its nodes is also a Fibonacci number. This analysis may be useful when addressing, for example, computational complexity (see [8] for a study on this topic with Fibonacci numbers).

2.1.2 Genealogical Tree

The problem can also be tackled using the Fibonacci identity $F_n = 1 + \sum_{i=1}^{n-2} F_i$. This decomposition is possible since the structure of the recursion tree for the recurrence relation is equivalent to the genealogical tree formed by the rabbit population growth process (see Fig. 1, where the numbers represent the months in which the rabbits are born). Replacing the summation with another function gives a new algorithm based on mutual recursion:

$$F_i = \begin{cases} 1 & \text{if } i = 1, 2 \\ 1 + H_{i-2} & \text{if } i \geq 3 \end{cases} \qquad H_i = \begin{cases} 0 & \text{if } i = 0 \\ F_i + H_{i-1} & \text{if } i \geq 1 \end{cases}$$

Nevertheless, it must be noted that mutual recursion does not appear naturally in this example as the result of performing problem decomposition.

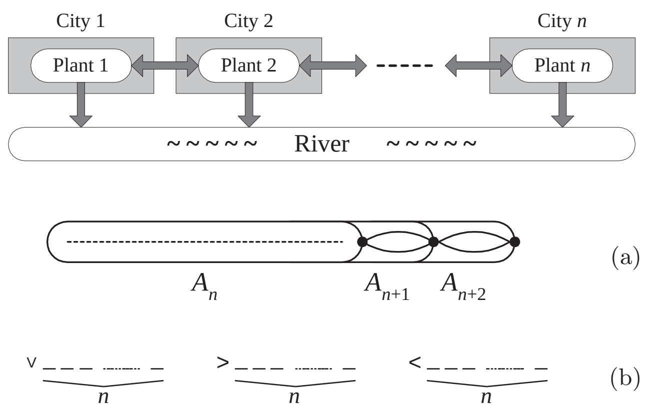


Figure 2: Water treatment plants puzzle, with direct (a) and mutual (b) recursive decompositions.

2.2 Water Treatment Plants Puzzle

In the "Water treatment plants puzzle" [3, 7] there are n cities, each with a water treatment plant, located along a side of a river (see Fig. 2). Each city generates sewage water that must be cleaned in a plant (not necessarily its own) and discharged into the river through pipes, where additional pipes connect neighboring cities. If a plant is working it will clean the sewage water from its city, plus any water coming from a pipe connecting a neighboring city, and discharge it to the river. However, a plant may not be working, in which case the water from its city, plus any water coming from a neighboring city, must be sent to another city. Given that water can flow in only one direction inside a pipe, the problem consists of determining the number of different ways it is possible to discharge the cleaned water into the river for n cities. It can be solved by modeling the water flow between cities, or the water discharge at each city, which has a more intuitive solution based on mutual recursion. In what follows, we will assume that the cities are arranged from left to right.

2.2.1 *Water Flow Between Cities*

For the $n-1$ pipes connecting the cities there are three possibilities denoted by the characters: ('N') if there is no flow; ('R') if water flows to the right; and ('L') if water flows to the left. In this situation the valid configurations involve strings of length $n-1$ where the substring "LR" does not appear, since it would mean that a city sends its water to more than one city.

Let A_n be the solution to the problem. It can be defined recursively as [3]:

$$A_{n+2} = 3A_{n+1} - A_n$$

which is a Fibonacci identity for even or odd sequence indices. For this problem, given the initial conditions ($A_1 = 1$ and $A_2 = 3$), $A_n = F_{2n}$. Despite the fact that the formula uses direct recursion, our classroom experience indicates that deriving it is rather tricky. Given the value of A_{n+1}, adding a new city increments the number of solutions to $3A_{n+1}$ (appending an 'N', 'R' or 'L'). However, the new strings that end in "LR" must not be counted, and the number of such solutions is exactly A_n (see Fig. 2a). Most students do not understand this subtraction easily.

2.2.2 *Water Discharge at Each City*

Another solution based on mutual recursion can be developed by modeling the direction in which the water is discharged at each city. Again, there are three possibilities: ('$\vee$') directly down to the river; ('$>$') to the city to the right; and ('$<$') to the city to the left. In this case, valid configurations are represented by strings of n characters, where the substring "$><$" is not permitted, nor a beginning with '$<$', nor an ending with '$>$'. According to these constraints, the problem can be decomposed into smaller similar subproblems. By fixing, for instance, the way in which the leftmost city discharges its water, the following three functions can be formulated (see Fig. 2b):

$$f_{\vee}(n) = \begin{cases} 2 & \text{if } n = 1 \\ f_{\vee}(n-1) + f_{>}(n-1) + f_{<}(n-1) & \text{if } n > 1 \end{cases}$$

$$f_{>}(n) = \begin{cases} 1 & \text{if } n = 1 \\ f_{\vee}(n-1) + f_{>}(n-1) & \text{if } n > 1 \end{cases}$$

$$f_{<}(n) = \begin{cases} 2 & \text{if } n = 1 \\ f_{\vee}(n-1) + f_{>}(n-1) + f_{<}(n-1) & \text{if } n > 1 \end{cases}$$

where, $f_{\vee}(n)$, $f_{>}(n)$, and $f_{<}(n)$ denote the total number of valid configurations for $n+1$ cities, assuming that the leftmost city discharges its water down to the river, to the right, or to the left, respectively (note that $f_{\vee}(n) = f_{<}(n)$). The solution is given by:

$$A_n = f_{\vee}(n-1) + f_{>}(n-1) = f_{>}(n)$$

Finally, it is easy to prove that $f_{\vee}(n) = F_{2n+1}$, and $f_{>}(n) = F_{2n}$. Hence, the solution is a Fibonacci number.

2.3 'Steven and Todd' Problem

The 'Steven and Todd' problem [6, 7] is an interesting example of how people with different educational backgrounds can provide completely different solutions or proofs (in this case mathematicians and computer scientists).

Consider the sequence $\alpha = \{a_1, a_2, \ldots, a_k\}$ of a subset of the first n positive integers that have been arranged in increasing order ($1 \leq a_1 < a_2 < \cdots < a_k \leq n$). The problem consists of finding the total number t_n of sequences α that start with an odd number and thereafter alternate in parity, that is,

$$\alpha = \{\text{odd}, \text{even}, \text{odd}, \text{even}, \ldots\}?$$

The solution to the problem is:

$$t_n = t_{n-1} + t_{n-2} = F_{n+2}$$

where $t_1 = 2$ for $\alpha = \emptyset, \{1\}$, and $t_2 = 3$ for $\alpha = \emptyset, \{1\}, \{1, 2\}$.

A formal proof for this problem can be found in [6]. This section describes an alternative - and perhaps simpler - approach to the problem, which consists of using an algorithm to construct each of the possible configurations, and then prove that the solution is F_{n+2}. For simplicity of implementation, however, an isomorphic problem will be addressed: calculating the number of sequences sequence $\delta = \{d_1, d_2, \ldots, d_k\}$ of a subset of the first n positive integers that have been arranged in *decreasing* order ($n \geq d_1 > d_2 > \cdots > d_k \geq 1$). In this case, the sequences start with an integer of the same parity as n, and thereafter alternate in parity. Note the equivalence between the δ and α sequences ($d_i = n + 1 - a_i$).

recursion type	linear	tail	multiple	mutual
function	A_{3418}	$B_{8,6}$	$C_{2,3}$	D_7
% of correct evaluations	76%	97%	83%	100%

Table 1: Percentages of correct evaluations for several function calls.

In this new setting, each δ can be generated from n to 1 with the following two functions:

$$Decide(i) = \begin{cases} 1 & \text{if } i < 1 \\ Name(i) + Decide(i-2) & \text{if } i \geq 1 \end{cases}$$

$$Name(i) = \begin{cases} 1 & \text{if } i = 1 \\ Decide(i-1) & \text{if } i > 1 \end{cases}$$

The process can be simulated by a simple game between two players (Steven and Todd), each naming a number lesser than the previous one, with the additional constraint that one player names odd numbers while the other even numbers. The logic behind the algorithm is straightforward. At each turn a player must decide between naming a number i, or deciding on the next possible integer to name $(i-2)$. After one player names a particular integer i, the turn switches to the second player, who must decide on the following integer $(i-1)$. It is trivial to see that $Decide(n) = F_{n+2}$, since $Name(n) = Decide(n-1)$. Moreover, it is not necessary to define the base case for $Name(n)$. Note that the algorithm returns the number of leafs of the recursion tree (all the valid sequences).

3. EXPERIMENTS AND RESULTS

A pedagogical evaluation was carried out to analyze the effectiveness of studying mutual recursion in more depth in a CS2 imperative programming course. It consisted of a written pre-test, followed by an identical post-test one month later. Recursion (in general) was taught in between the tests, where the problems mentioned in this paper were used to study mutual recursion. The test included items associated with the first five levels of Bloom's taxonomy, and with different types of recursion: linear (non-tail), tail, multiple, mutual (and nested). A total of 58 students participated in the evaluation (21 second year, 33 third year, 3 fourth year, and 1 fifth year student), all with previous knowledge of recursion.

Regarding definitions, linear, tail, multiple and mutual recursion were described correctly in the post-test by 62%, 60%, 83%, and 86% of the students, respectively. As can be seen, mutual recursion was defined better than other types of recursion.

The test also included items associated with calculating values of the following recursive mathematical functions (defined over nonnegative integers):

$$A_n = \begin{cases} n\%2 & \text{if } 0 \leq n < 10 \\ ((n\%10)\%2) + A_{n/10} & \text{if } n \geq 10 \end{cases}$$

$$B_{n,m} = \begin{cases} n & \text{if } m = 0 \\ B_{m,n\%m} & \text{if } m > 0 \end{cases}$$

$$C_{n,m} = \begin{cases} 0 & \text{if } n = 0 \text{ or } m = 0 \\ (C_{n-1,m} + C_{n,m-1} + n + m)/2 & \text{if } n > 0 \text{ and } m > 0 \end{cases}$$

$$D_n = \begin{cases} 1 & \text{if } n = 0 \\ E_{n/2} & \text{if } n > 0 \end{cases} \quad E_n = \begin{cases} 0 & \text{if } n = 0 \\ D_{n-1} & \text{if } n > 0 \end{cases}$$

problem (4)	mutual	linear
# correct implementations	26	6
# incorrect implementations	9	12

Table 2: Contingency table showing the relationship between the type of recursion used and the number of correct implementations in problem (4).

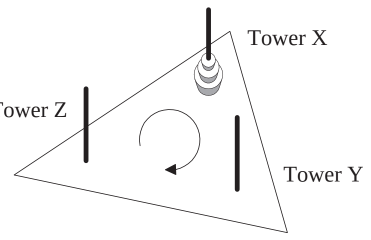
where % and / represent the remainder and quotient of integer divisions, respectively. Table 1 shows the percentages of correct evaluations in the post-test for several function calls. The difficulty appears to lie in the number and complexity of the mathematical operations to be carried out, rather than in the sequence of operations that will lead to a particular solution, which is claimed to account for much of the complexity associated with mutual recursion. Note that the percentages are higher for tail recursive functions. Additionally, every student obtained a correct answer for the mutual recursion case.

A major goal was to evaluate the students' abilities to design recursive algorithms. Students were asked to implement four algorithms: (1) the quotient of an integer division using linear (non-tail) recursion, (2) the same quotient using tail recursion, (3) a combinatorial coefficient, and (4) the solution to the following problem:

Two employees, A and B, work in a warehouse which must be emptied of parcels. They take turns to remove the parcels using a different strategy: employee A always removes one parcel, while employee B removes two parcels if the number of parcels left is even, and one if it is odd. If employee A always removes the first parcel, calculate the number of turns needed to empty a warehouse containing n parcels.

The percentages of correct implementations in the post-test for the four problems were 50% (1), 36% (2), 76% (3), and 55% (4). Additionally, the fourth problem can be implemented through mutual or linear recursion. Table 2 shows the relationship between the type of recursion used and the number of correct implementations for problem (4), where five students did not provide any solution. These results suggest that students do not have more difficulty in understanding and applying mutual recursion than direct recursion. Rather, the complexity arises from the formulation of the problem. Furthermore, not only did they prefer mutual recursion over linear on the fourth problem (35 vs. 18), but also 74% of the implementations based on mutual recursion were correct, while only 33% of the solutions based on linear recursion had a proper implementation.

Comparing the type of recursion used for problem (4) in the pre-test vs. the post-test, it is worth mentioning that 10 students switched from implementing a solution based on linear recursion in the pre-test to coding a mutually recursive algorithm in the post test. None of these 10 students had a correct solution in the pre-test, but seven designed an appropriate implementation in the post-test. Furthermore, students who did not provide any solution in the pre-test did not prefer any particular type of recursion (six mutual and six linear solutions). However, out of these 12 students, three implemented a correct solution in the post-test by using mutual recursion, while only one arrived at a proper implementation based on linear recursion.



```
Clock(n,X,Y,Z)
 if(n>0)
  AntiClock(n-1,X,Z,Y)
  Move disk n from X to Y
  AntiClock(n-1,Z,Y,X)
```

```
AntiClock(n,X,Z,Y)
 if(n>0)
  AntiClock(n-1,X,Z,Y)
  Move disk n from X to Y
  Clock(n-1,Z,X,Y)
  Move disk n from Y to Z
  AntiClock(n-1,X,Z,Y)
```

Figure 3: Algorithm for solving the cyclic towers of Hanoi problem.

4. DISCUSSION AND CONCLUSIONS

This paper provided a collection of problems with mutually recursive solutions that are derived through induction, and can be understood by CS2 students without any previous exposure to functional programming. Emphasis is on problem decomposition, where it is not necessary, or even desirable, to analyze recursion trees (this may be more suitable when explaining more advanced concepts, such as backtracking, memoization, computational complexity, etc.).

The experiments indicate that students did not find difficulties in evaluating simple mutually recursive mathematical functions. Therefore, determining the sequence of operations that will lead to a particular solution is not necessarily more difficult for mutual recursion. Furthermore, the ability to analyze the structure of the recursion tree, and evaluate the result of the various recursive calls, is not necessarily related to the ability to decompose problems in a recursive manner. Studies show that the key to understanding functional abstraction is "*not* to think too hard" [12], in the sense that students must focus on *what* the algorithm does, rather than on *how* it does it.

The article reports several results related to learning mutual recursion (the authors are not aware of any other study involving an experimental evaluation focused on this topic). Our conclusion is that mutual recursion is not as hard as it seems. The chances of obtaining a correct solution increased when using mutual recursion in problem (4), see Sec. 3. This reflects the fact that a problem may have a more intuitive mutually recursive solution, depending on how it is expressed. Therefore, it is important that students be exposed to mutual recursion at some point in their curricula. A natural place would be a functional programming course. However, results indicate that students assimilate mutual recursion well in a CS2 imperative programming course.

There exist other problems (besides combinatorial) with mutually recursive solutions that are useful for teaching recursion. One example is the 'Cyclic Towers of Hanoi' [1], an elegant and illustrative variant of the Towers of Hanoi problem, where the disks must move in only one direction. Figure 3 shows the solution to move n disks from tower X to Y using Z in clockwise direction, and from X to Z using Y in anticlockwise direction, where every single disk moves in clockwise direction. It constitutes an ideal companion to the original problem since the fundamental reasoning about the movement of the disks, as well as the decomposition of the problem into sub-problems, remain basically the same.

We believe that the proposed algorithms provide an alternative and enlightening point of view of recursion, which can enhance students' comprehension and assimilation of problem decomposition, functional abstraction and induction.

Finally, the problems discussed in this paper, along with other algorithmic or combinatorial problems, can be easily used in a problem-based learning [5] framework.

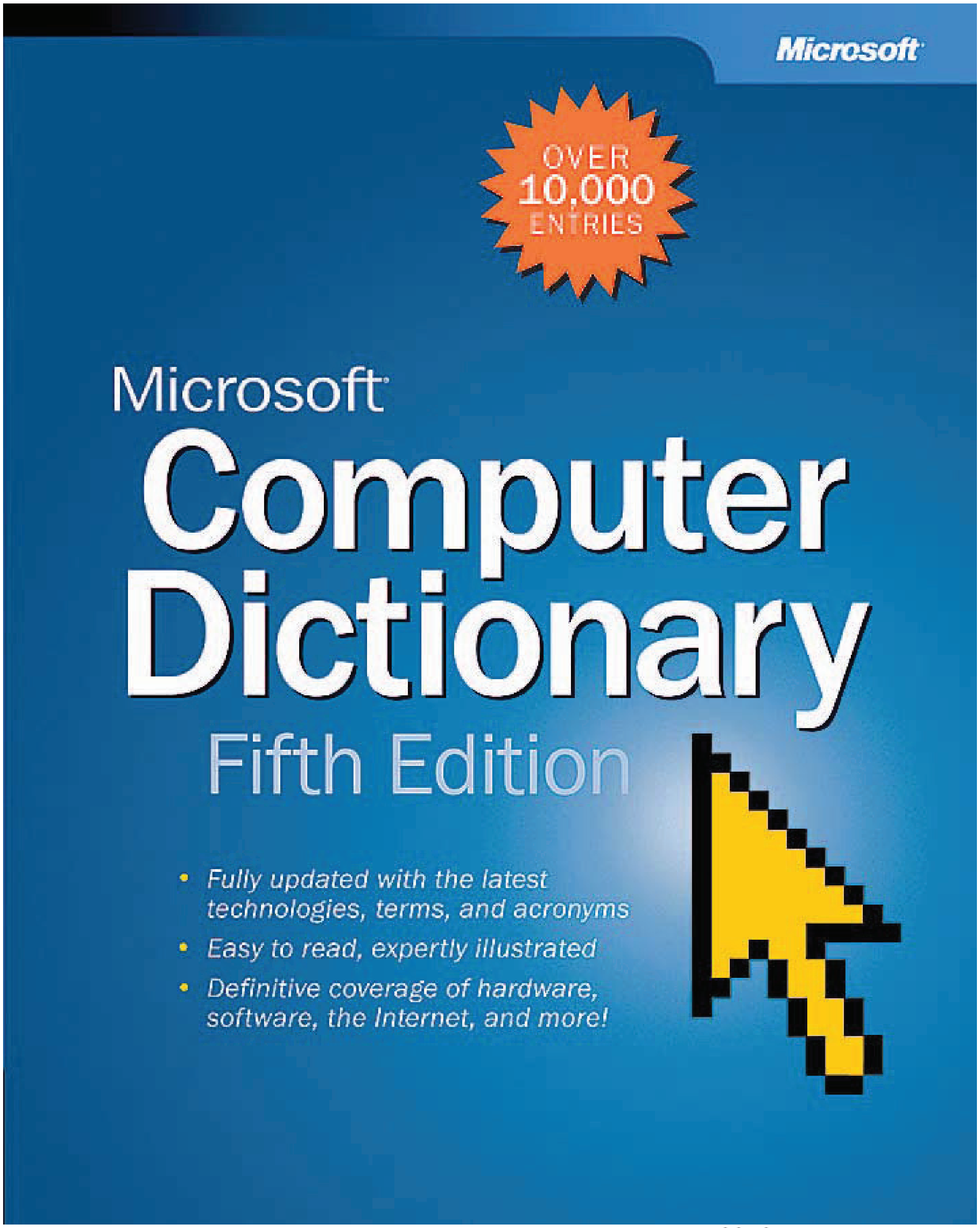
5. ACKNOWLEDGEMENTS

Research partially supported by the Spanish MEC projects: WEST/FAST TIN2006-15578-C02-01, and TIN2004-07568. We would like to thank Dr. J. A. Velázquez-Iturbide, as well as the anonymous reviewers, for their helpful comments.

6. REFERENCES

[1] M. D. Atkinson. The cyclic towers of Hanoi. *Information Processing Letters*, 13(3):118–119, 1981.
[2] W. Dann, S. Cooper, and R. Pausch. Using visualization to teach novices recursion. *SIGCSE Bull.*, 33(3):109–112, 2001.
[3] R. A. Deininger. Fibonacci numbers and water pollution control. *The Fibonacci Quarterly*, 10(3):299–302, 1972.
[4] S. M. Haynes. Explaining recursion to the unsophisticated. *SIGCSE Bull.*, 27(3):3–6, 1995.
[5] C. E. Hmelo-Silver. Problem-based learning: What and how do students learn? *Educational Psychology Review*, 16(3):235–266, 2004.
[6] R. Honsberger. *Mathematical Chestnuts from Around The World*. Mathematical Association of America, 2001. Sect. 17.
[7] R. Knott. Fibonacci numbers and the golden section, 1996-2008. www.mcs.surrey.ac.uk/Personal/R.Knott/Fibonacci/fib.html.
[8] O. Martín-Sánchez and C. Pareja-Flores. A gentle introduction to algorithm complexity for cs1 with nine variations on a theme by Fibonacci. *SIGCSE Bull.*, 27(2):49–56, 1995.
[9] C. Pareja-Flores, J.Urquiza-Fuentes, and J. A. Velázquez-Iturbide. WinHIPE: an IDE for functional programming based on rewriting and visualization. *SIGPLAN Not.*, 42(3):14–23, 2007.
[10] M. Rubio and B. Pajak. Fibonacci numbers using mutual recursion. In *Proc. of the Fifth Finnish/Baltic Sea Conference on Computer Science Education*, 41, pages 174–177, 2005.
[11] M. Rubio-Sánchez and I. Hernán-Losada. Exploring recursion with Fibonacci numbers. *SIGCSE Bull.*, 39(3):359, 2007.
[12] R. Sooriamurthi. Problems in comprehending recursion and suggested solutions. *SIGCSE Bull.*, 33(3):25–28, 2001.
[13] J. A. Velázquez-Iturbide. Recursion in gradual steps (is recursion really that difficult?). *SIGCSE Bull.*, 32(1):310–314, 2000.
[14] C. C. Wu, N. B. D., and L. J. Bethel. Conceptual models and cognitive learning styles in teaching recursion. *SIGCSE Bull.*, 30(1):292–296, 1998.

APPENDIX E



Microsoft

OVER 10,000 ENTRIES

Microsoft

Computer Dictionary

Fifth Edition

- Fully updated with the latest technologies, terms, and acronyms
- Easy to read, expertly illustrated
- Definitive coverage of hardware, software, the Internet, and more!

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Cataloging-in-Publication Data
Microsoft Computer Dictionary.--5th ed.
p. cm.
ISBN 0-7356-1495-4
1. Computers--Dictionaries. 2. Microcomputers--Dictionaries.

AQ76.5. M52267 2002
004'.03--dc21 200219714

Printed and bound in the United States of America.

2 3 4 5 6 7 8 9 QWT 7 6 5 4 3 2

Distributed in Canada by H.B. Fenn and Company Ltd.

A CIP catalogue record for this book is available from the British Library.

Acquisitions Editor: Alex Blanton
Project Editor: Sandra Haynes

Body Part No. X08-41929

the exponent (E+05) shows the power of 10 to which 6.4 is raised. *Also called:* significand. *See also* floating-point notation.

manual link *n.* A link that requires you to take action to update your data after the data in the source document changes.

many-to-many relationship *n.* A complex association between two sets of parameters in which many parameters of each set can relate to many others in the second set. A many-to-many relationship is most commonly used to describe an association between two tables in which one record in either table can relate to many records in the other table.

many-to-one relationship *n.* **1.** A server configuration in which several small servers replicate the abilities of one larger, more powerful server. *See also* key pair. **2.** In reference to asymmetric key encryption, the idea that many individuals in possession of the public key can decrypt the digital signature of one individual in possession of the private key.

map¹ *n.* Any representation of the structure of an object. For example, a memory map describes the layout of objects in an area of memory, and a symbol map lists the associations between symbol names and memory addresses in a program. *See also* image map.

map² *vb.* To translate one value into another. For example, in computer graphics one might map a three-dimensional image onto a sphere. In reference to virtual memory systems, a computer might translate (map) a virtual address into a physical address. *See also* virtual memory.

MAPI *n.* Acronym for **M**essaging **A**pplication **P**rogramming **I**nterface. The Microsoft interface specification that allows different messaging and workgroup applications (including e-mail, voice mail, and fax) to work through a single client, such as the Exchange client included with Windows 95 and Windows NT. *See also* application programming interface.

mapped data field *n.* A field that represents commonly used information, such as “First Name.” If a data source contains a “First Name” field or variation, such as “FName,” the data source field automatically maps to the corresponding mapped data field.

mapped drives *n.* **1.** In the Windows environment, network drives that have been assigned local drive letters and are locally accessible. **2.** Under UNIX, disk drives that have been defined to the system and can be made active.

MapPoint *n.* Business mapping software introduced by Microsoft as an Office-compatible product in 1999. Designed for use by business people, MapPoint consists of a database of United States maps showing detail down to the level of individual streets and demographic data broken out by state, county, zip code, and other regions. *See also* Office.

margin *n.* In printing, those portions of a page—top, bottom, and sides—outside the main body of text.

mark *n.* **1.** In applications and data storage, a symbol or other device used to distinguish one item from others like it. **2.** In digital transmission, the state of a communications line (positive or negative) corresponding to a binary 1. In asynchronous serial communications, a mark condition is the continuous transmission of binary 1s to indicate when the line is idle (not carrying information). In asynchronous error checking, setting the parity bit to 1 in each group of transmitted bits is known as mark parity. *See also* parity. *Compare* space. **3.** In optical sensing, a pencil line, as on a voting form or an IQ test, that can be recognized by an optical reader.

marker *n.* **1.** Part of a data communications signal that enables the communications equipment to recognize the structure of the message. Examples are the start and stop bits that frame a byte in asynchronous serial communications. **2.** A symbol that indicates a particular location on a display surface.

Mark I *n.* **1.** An electromechanical calculating machine designed in the late 1930s and early 1940s by Howard Aiken of Harvard University and built by IBM. *Also called:* Automatic Sequence Controlled Calculator, Harvard Mark I. **2.** The first fully electronic stored-program computer, designed and built at Manchester University in England. It successfully executed its first program in June 1948. **3.** The first commercial computer, which was based on the Manchester Mark I and released in 1951.

markup *n.* Comments and tracked changes such as insertions, deletions, and formatting changes that you can view or print.

markup language *n.* A set of codes in a text file that instructs a computer how to format the file on a printer or video display or how to index and link its contents. Examples of markup languages are Hypertext Markup Language (HTML) and Extensible Markup Language (XML), which are used in Web pages, and Standard Generalized Markup Language (SGML), which is used for typesetting and desktop publishing purposes and in electronic documents. Markup languages of this sort are designed to enable documents and other files to be platform-independent and highly portable between applications. *See also* HTML, SGML, XML.

marquee *n.* A nonstandard HTML extension that causes scrolling text to appear as part of a Web page. Currently, marquees are viewable only with Internet Explorer. *See also* HTML, Internet Explorer, Web page.

marquee component *n.* A region on a page that displays a horizontally scrolling text message.

mask *n.* **1.** A binary value used to selectively screen out or let through certain bits in a data value. Masking is performed by using a logical operator (AND, OR, XOR, or NOT) to combine the mask and the data value. For example, the mask 00111111, when used with the AND operator, removes (masks off) the two uppermost bits in a data value but does not affect the rest of the value. See the illustration. *See also* logical operator, mask bit. **2.** In television and display technology, a thin perforated sheet of metal or a close-set series of metal strips on the surface of the screen that helps create a clear, sharp image by ensuring that the electron beam for a particular color (red, blue, or green) strikes only the phosphor it is intended to illuminate, while the phosphors for the other colors are shadowed by the mask. Three types of masks are in use: a shadow mask, with round perforations; an aperture grill, with vertical stripes; and a slot mask, with elliptical openings. *See also* aperture mask, shadow mask, slot mask.

11010101	Data value
AND 00111111	Mask
00010101	Resulting value

Mask.

maskable interrupt *n.* A hardware interrupt that can be temporarily disabled (masked) during periods when a program needs the full attention of the microprocessor. *See also* external interrupt, hardware interrupt, interrupt. *Compare* nonmaskable interrupt.

mask bit *n.* A given bit within a binary mask whose function is to screen out or let through the corresponding bit in a data value when the mask is used in an expression with a logical operator. *See also* mask (definition 1).

masking *n.* The process of using the *mask* operation to perform operations on bits, bytes, or words of data. *See also* mask (definition 1).

mask off *vb.* To use a mask to remove bits from a byte of data. *See also* mask (definition 1).

massively parallel processing *n.* A computer architecture in which each of a large number of processors has its own RAM, which contains a copy of the operating system, a copy of the application code, and its own part of the data, on which that processor works independently of the others. *Acronym:* MPP. *Compare* SMP.

massively parallel processor *n.* A computer designed to perform massively parallel processing.

mass storage *n.* A generic term for disk, tape, or optical disc storage of computer data, so called for the large masses of data that can be stored in comparison with computer memory capacity. *Compare* memory.

Master Boot Record *n.* The first sector of the first hard disk; a physically small but critical element in the startup process on an x86-based computer. When a computer is booted, it processes a series of self-tests and then reads the Master Boot Record, or MBR, into memory. The MBR contains instructions that locate the disk's system (startup) partition, read the contents of the first sector of the system partition into memory, and then carry out the instructions contained in that sector. If the sector represents what is known as a Partition Boot Sector, the instructions found there begin the process of loading and starting the operating system. In other words, the startup process on an x86-based computer is as follows: self-test to Master Boot Record; MBR to system partition and Partition Boot Sector; Partition Boot Sector to operating system; and, finally, a computer ready to go to work. *Acronym:* MBR. *See also* Partition Boot Sector.

master file *n.* In a set of database files, the file containing more or less permanent descriptive information about the principal subjects of the database, summary data, and one or more critical key fields. For example, customers' names, account numbers, addresses, and credit terms might be stored in a master file. *See also* master record. *Compare* transaction file.

master key *n.* The server-based component of software or data protection. In some systems, data or applications are stored on a server and must be downloaded to the local machine for use. When a client requests the data, it presents a session key. If the session key supplied matches the master key, the key server sends the requested packet. *See also* client (definition 3), server (definition 2).

M

tag *n.* **1.** In programming, one or more characters containing information about a file, record type, or other structure. **2.** In certain types of data files, a key or an address that identifies a record and its storage location in another file. *See also* tag sort. **3.** In markup languages such as SGML and HTML, a code that identifies an element in a document, such as a heading or a paragraph, for the purposes of formatting, indexing, and linking information in the document. In both SGML and HTML, a tag is generally a pair of angle brackets that contain one or more letters and numbers. Usually one pair of angle brackets is placed before an element, and another pair is placed after, to indicate where the element begins and ends. For example, in HTML, <I>hello world</I> indicates that the phrase "hello world" should be italicized. *See also* <>, element, emotag, HTML, SGML. **4.** An early-generation raster graphics format used for Macintosh Ready, Set, Go programs and Letraset's ImageStudio. *See also* raster graphics.

Tagged Image File Format *n. See* TIFF.

tag sort *n.* A sort performed on one or several key fields for the purpose of establishing the order of their associated records. *Also called:* key sort.

tag switching *n.* A multilayer Internet switching technology developed by Cisco Systems that integrates routing and switching.

talk^1 *n.* The UNIX command that, when followed by another user's name and address, is used to generate a request for a synchronous chat session on the Internet. *See also* chat^1 (definition 1).

talk^2 *vb. See* chat^2.

talker *n.* An Internet-based synchronous communication mechanism most commonly used to support multiuser chat functions. Such systems typically provide specific commands for movement through separate *rooms*, or chat areas, and allow users to communicate with other users in real time through text messages, indicate simple gestures, use a bulletin board system (BBS) for posting comments, and send internal e-mail. *See also* BBS (definition 1), chat^1 (definition 1).

talk. newsgroups *n.* Usenet newsgroups that are part of the talk. hierarchy and have the prefix talk. as part of their names. These newsgroups are devoted to debate and discussion of controversial topics. Talk. newsgroups are one of the seven original Usenet newsgroup hierarchies. The other six are comp., misc., news., rec., sci., and soc. *See also* newsgroup, traditional newsgroup hierarchy, Usenet.

tandem processors *n.* Multiple processors wired so that the failure of one processor transfers central processing unit (CPU) operation to another processor. Using tandem processors is part of the strategy for implementing fault-tolerant computer systems. *See also* central processing unit.

TANSTAAFL *n.* Acronym for **T**here **a**in't **n**o **s**uch **t**hing **a**s **a** **f**ree **l**unch. An expression used on the Internet in e-mail, chat sessions, mailing lists, newsgroups, and other online forums; derived from *The Moon Is a Harsh Mistress*, a science-fiction classic by Robert A. Heinlein. *See also* chat^1 (definition 1), e-mail^1 (definition 1), mailing list, newsgroup.

tap^1 *n.* A device that can be attached to an Ethernet bus to enable a computer to be connected.

tap^2 *vb.* To use a stylus to quickly touch a device screen to perform an activity. Tapping is analogous to clicking with a mouse.

tap and hold *vb.* To hold a stylus on a device screen to open a pop-up or shortcut menu. Analogous to right-clicking with a mouse.

tape *n.* **1.** A thin strip of polyester film coated with magnetic material that permits the recording of data. Because tape is a continuous length of data storage material and because the read/write head cannot "jump" to a desired point on the tape without the tape first being advanced to that point, tape must be read or written sequentially, not randomly (as can be done on a floppy disk or a hard disk). **2.** A storage medium consisting of a thin strip of paper used to store information in the form of sequences of punched holes, chemical impregnation, or magnetic ink imprinting.

tape cartridge *n.* A module that resembles an audio cassette and contains magnetic tape that can be written on and read from by a tape drive. Tape cartridges are primarily used to back up hard disks. *See also* tape (definition 1).

tape drive *n.* A device for reading and writing tapes. *See also* tape (definition 1).

tape dump *n.* The process of simply printing the data contained on a tape cartridge without performing any report formatting. *See also* tape cartridge.

tape tree *n.* A means of audiotape distribution, used in Usenet music newsgroups and mailing lists, in which a

ISO/OSI MODEL	
ISO/OSI Layer	***Focus***
Application *(highest level)*	Program-to-program transfer of information
Presentation	Text formatting and display, code conversion
Session	Establishing, maintaining, and coordinating communication
Transport	Accurate delivery, service quality
Network	Transport routes, message handling and transfer
Data-link	Coding, addressing, and transmitting information
Physical	Hardware connections

Transport layer.

Transport Layer Security *n. See* TLS.

transpose¹ *n.* The result of rotating a matrix.

transpose² *vb.* **1.** To reverse, as the order of the letters *h* and *t* in *hte*, in correcting the spelling of *the*; or reversing two wires in a circuit. **2.** In mathematics and spreadsheets, to rotate a matrix (a rectangular array of numbers) about a diagonal axis.

transputer *n.* Short for **trans**istor com**puter**. A complete computer on a single chip, including RAM and an FPU, designed as a building block for parallel computing systems.

trap¹ *n. See* interrupt.

trap² *vb.* **1.** To intercept an action or event before it occurs, usually in order to do something else. Trapping is commonly used by debuggers to allow interruption of program execution at a given spot. *See also* interrupt, interrupt handler. **2.** To slightly overlap adjacent colors in preparing material for printing. Page layout and prepress programs trap color to prevent gaps between colors caused by minor variations in registration during printing.

trapdoor *n. See* back door.

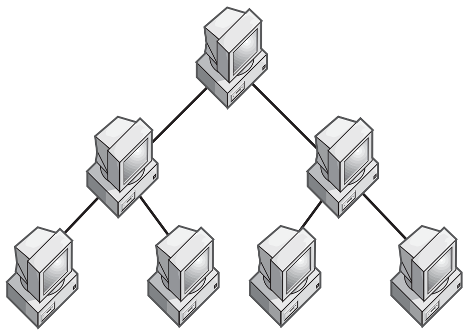
trap handler *n. See* interrupt handler.

Trash *n.* An icon on the screen in the Macintosh Finder, resembling a garbage can. To delete a file or eject a diskette, the user drags the icon for the file or diskette to the Trash. However, until the user shuts down the system or chooses the menu option "Empty Trash," a file in the Trash is not actually deleted; the user can retrieve it by double-clicking the Trash icon and dragging the file's icon out of the resulting window. *Compare* Recycle Bin.

traverse *vb.* In programming, to access in a particular order all of the nodes of a tree or similar data structure.

tree *n.* A data structure containing zero or more nodes that are linked together in a hierarchical fashion. If there are any nodes, one node is the root; each node except the root is the child of one and only one other node; and each node has zero or more nodes as children. *See also* child (definition 2), graph, leaf, node (definition 3), parent/child (definition 2), root.

tree network *n.* A topology for a local area network (LAN) in which one machine is connected to one or more other machines, each of which is connected to one or more others, and so on, so that the structure formed by the network resembles that of a tree. See the illustration. *See also* bus network, distributed network, ring network, star network, token ring network, topology.



Tree network.

tree search *n.* A search procedure performed on a tree data structure. At each step of the search, a tree search is able to determine, by the value in a particular node, which branches of the tree to eliminate, without searching those branches themselves. *See also* branch (definition 1), tree structure.

tree structure *n.* Any structure that has the essential organizational properties of a tree. *See also* tree.

tree view *n.* A hierarchical representation of the folders, files, disk drives, and other resources connected to a computer or network. For example, Windows Explorer uses a tree view to display the resources that are attached to a computer or a network. *See also* resource.